쇼핑몰 AWS Migration

아키텍처 설계

Suyeon Shoes

2021년 05월 21일

—

목차

1. 프로젝트 개요

1.1. 프로젝트 명 및 기간

프로젝트 명 : 쇼핑몰 AWS Migration 아키텍처 설계

프로젝트 기간 : 2021.04.20 ~ 2021.05.21

1.2. 프로젝트 배경 및 목표

Solution Architect의 관점으로 최적의 Solution을 도출하는 역량을 습득하는 것이 프로젝트의 주 목적이다. 5개월 간 AWS 클라우드 인재 양성 훈련을 통해 학습한 내용을 바탕으로 IDC 상의 On-Premise를 사용중인 쇼핑몰 및 스타트업 회사 상대로 AWS Migration을 제안하는 것이 본 프로젝트의 목표이다.

1.2.1. 프로젝트 배경

최근 가장 큰 이슈인 COVID-19로 인한 사회적 거리 두기 영향으로 고객 소비 패턴이 오프라인에서 온라인으로 급변함에 따라 오프라인 매장이 축소되었다. 그 결과 온라인 플랫폼 확장이 필요한 기업들은 빠른 전환 및 확장을 하지 못해 온라인 시장 진입이 늦어져 매출 감소 등 피해를 입었다. 때문에 기업들은 확장성과 가용성이 높은 클라우드 환경으로 Migration하는 추세이다. 이런 배경을 바탕으로 온라인 플랫폼으로 확장하기 위한 기업을 대상으로 프로젝트를 진행하게 되었다.

1.2.2. 목표

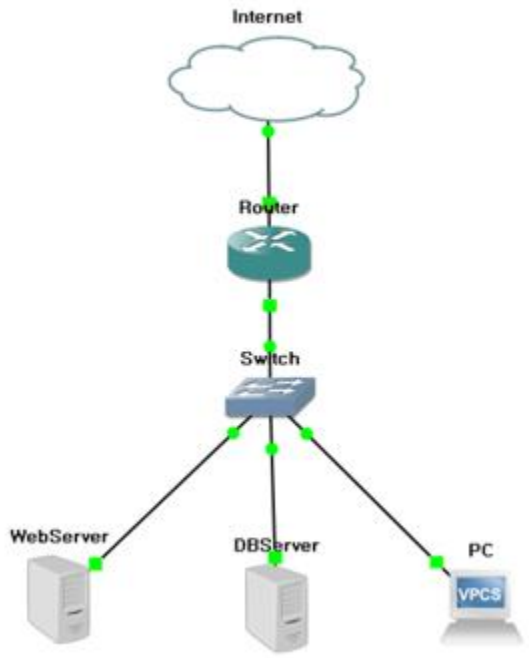
On-Premise를 사용중인 기업을 선정했다. 컨설팅 업체의 입장에서 쇼핑몰을 대상으로 최적의 AWS Migration 솔루션을 제안하고, 빠른 개발환경 구축 및 재해복구 환경을 제공하는 것이 본 프로젝트의 목표이다.

1.3. 프로젝트 설명

본 프로젝트는 On-Premise 환경을 AWS 클라우드로 Migration 하는 것과 효율적인 자원 사용 및 비용 절감을 위해 신축성과 유연성에 초점을 맞춘 클라우드 환경을 구축하고자 한다. 또한 기업 내 조직 간 경계를 두어 내부 리소스의 접근 권 한에 대한 보안성을 보장하며 재해 상황 발생 시 빠른 시간 내에 다시 서비스를 제공할 수 있도록 복구 시나리오를 포함하여 기업의 서비스 제공에 신뢰성을 높이는 것을 목적으로 한다.

1.3.1. On-Premise

마이그레이션 대상의 On-Premise 환경은 [그림 1]의 구성도와 같다. Web Server 1대와 DB Server 1대로 구성되어 있다. 이러한 환경은 확장성 및 가용성과 비용에 대한 문제가 발생할 수 있다.

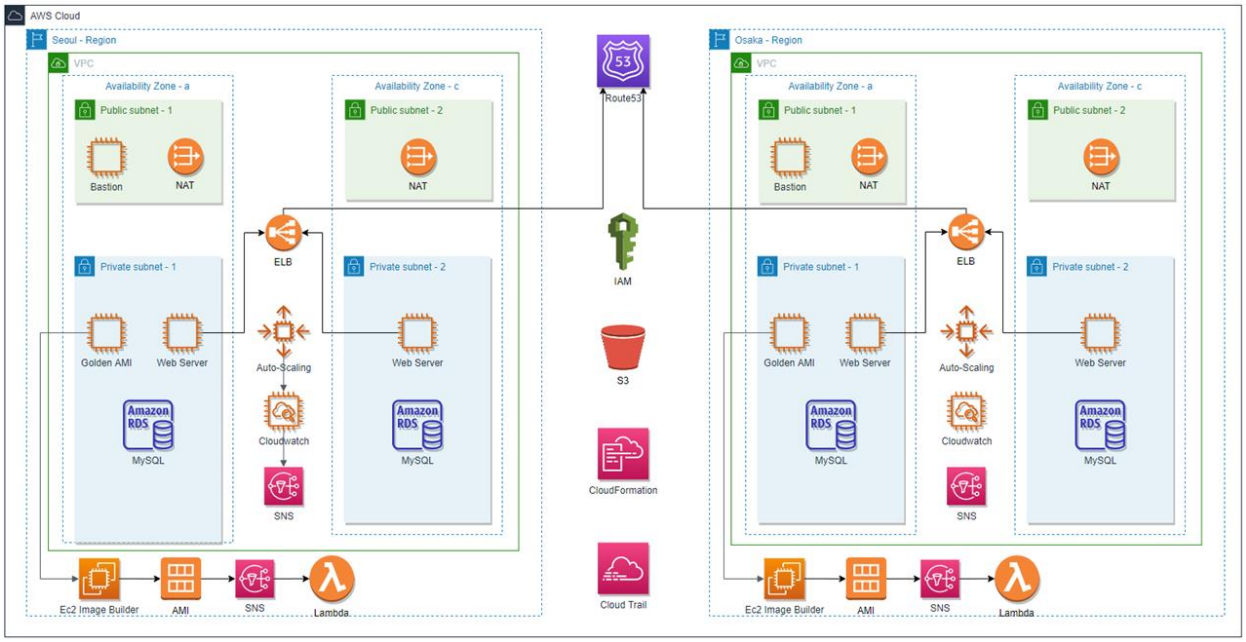


[그림 1] On-Premise 환경 구성

1.4. 사용 프로그램

AWS Services	IAM, EC2, RDS, VPC, CloudWatch, Cloudformation, S3, SNS, ELB, Auto Scaling, Route53, EC2 Image Builder, Lambda, CloudTrail
OS	Ubuntu 18.04, Windows 10
원격 접속 소프트웨어	Xshell 7
스크립트 언어	YAML, Python
웹 서버	PHP, HTML, CSS, JavaScript, Apache2
데이터베이스	Mysql 8.0.20
부하 테스트 도구	Locust
문서 프로그램	Google Docs
버전 툴 킷	Git Hub

1.5. AWS 설계



[그림 3] AWS 환경 다이어그램

본 프로젝트는 Seoul Region과 Osaka Region에 VPC가 각각 구성된다. 먼저 Seoul Region에는 고가용성을 위해 가용 영역 ap-northeast-2a와 ap-northeast-2c를 사용하고, Public Subnet과 Private Subnet이 각각의 가용 영역에 있으며, Public Subnet에는 BastionHost와 Nat Instance가, Private Subnet에는 Web Server와 RDS가 생성된다. 이때 Private Subnet 1에는 AMI 이미지 관리를 위해 Golden AMI를 생성하여 사용하게 된다. 이미지 배포에는 EC2 Image Builder를 사용한다. EC2 Image Builder를 통하여 이미지 배포가 완료되면 SNS를 통해 Lambda를 이용해 Auto Scaling의 새로고침을 통하여 롤링배포를 한다. 사용자는 서비스가 업데이트 또는 수정이 되더라도 서비스 중단 없이 계속 사용할 수 있도록 환경을 구성하였다.

Osaka Region 또한 Seoul Region과 동일한 인프라를 가진다.

2. 기업 내 IAM(Identity and Access Management) USER

2.1. Master Account

Root 권한 대행 유저로서 서버가 다운됐을 시 재해복구 시스템을 이용하여 다른 지역에 서버를 올림으로서 신속한 대응처리를 하고 로그 기록을 관리하여 업무 수행 또는 오류 등을 검토한다. 또한 Billing을 체크하면서 AWS Cloud 이용에 대하여 관리한다.

2.2. Webserver User

Webserver User는 Seoul Region과 Osaka Region의 Public Subnet에 위치한 Bastion Host의 접근 권한 및 Private Subnet에 위치한 Web Server에 대한 권한을 가지게 된다. 해당 User는 Bastion Host를 통해 Golden AMI에 접속하여 Git에 작성된 Web Server Code를 불러와 관리하게 된다.

2.3. Database User

Database User는 Seoul Region과 Osaka Region의 Public Subnet에 위치한 Bastion Host 에 대한 접근 권한 및 Private Subnet에 위치한 RDS에 대한 권한을 가지게 된다. 각 지역의 Private에 위치해 있는 Database의 데이터(회원 정보, 제품 정보 등)를 관리하고 수집한다.

2.4. Network User

Network User는 전반적인 인프라 구성을 관리 및 유지한다. 전반적인 인프라 라고 함은 Seoul Region 과 Osaka Region의 VPC, Subnet 등을 의미한다.

2.5. Developer User

Developer User는 EC2 접근 권한을 가지고 있으며, 쇼핑몰에서 진행 중 및 향후 필요한 개발에 관하여 요구되는 Resources를 업데이트하고 관리한다.

3. 웹 서버 개발

3.1. 쇼핑몰 서버

신발 쇼핑몰은 PHP을 기반으로 구축했으며, Front End의 페이지는 HTML,CSS, JavaScript를 사용하여 구성했다. 소비자의 입장에서 주로 상호작용하는 서버이므로 다음과 같이 구성 및 기능들을 만들었다.

- Web page
- 소비자 회원가입 및 로그인
- 국가 별 사이트 이동
- 고정 세션을 사용한 데이터 유지
- 물품 구매

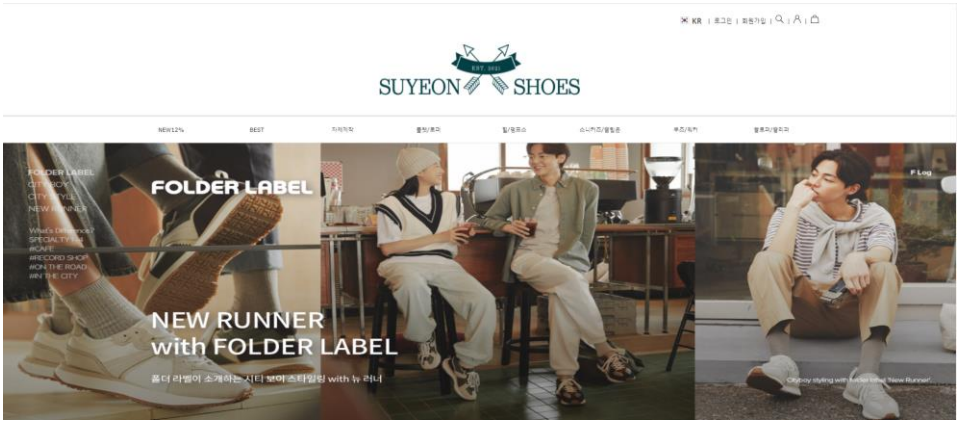
신발 쇼핑몰에서 사용하고 있는 AWS 서비스들은 다음과 같다

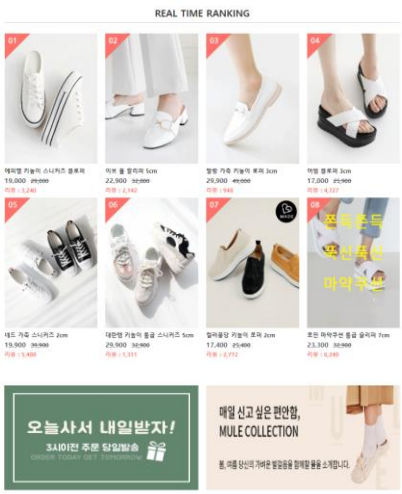
- VPC
- EC2
- RDS
- ALB
- SNS
- Cloud Watch
- Auto Scaling

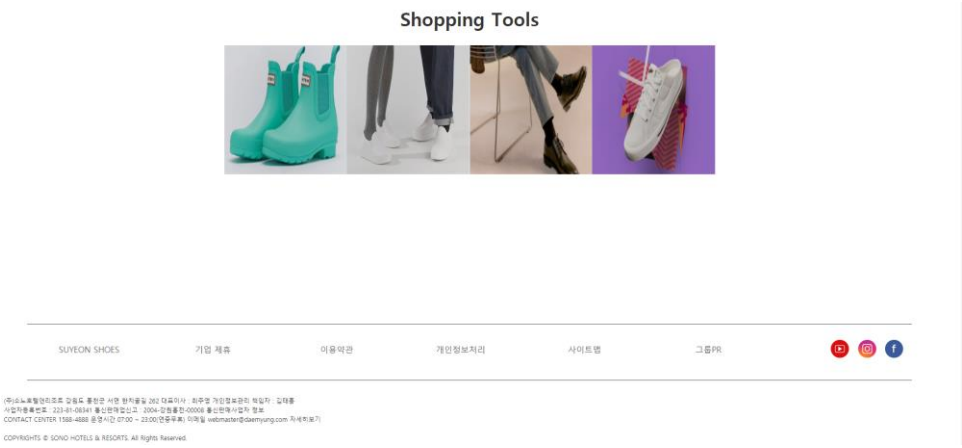
- Lambda
- EC2 Image Builder
- Route 53
- S3
- Cloud Formation

3.1.1. 웹 페이지

소비자는 쇼핑몰 접속 시 Suyeon Shoes의 메인 홈페이지**[그림 1]**에 접속하게 된다. 메인 홈페이지 에서는 회원가입 및 로그인 페이지로 이동 가능하며, Japan-Osaka 홈페이지로도 이동 가능하다.





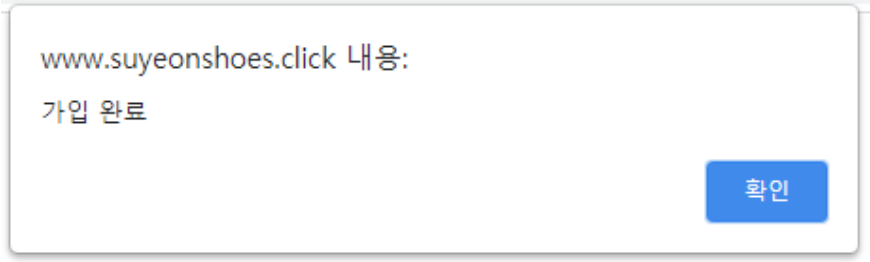


[그림5] Main Page

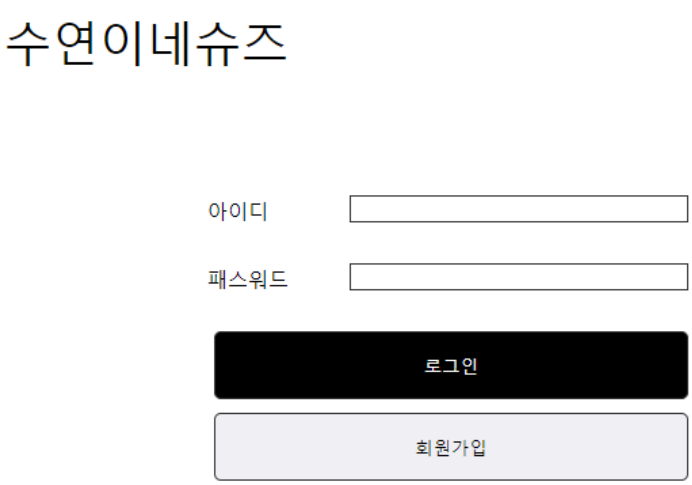
3.1.2. 회원가입 및 로그인

소비자는 홈페이지에 있는 상품에 대한 정보를 회원가입 없이 해당 상품의 정보를 볼 수 있지만 구매를 하기 위해서는 회원가입과 로그인을 해야한다.

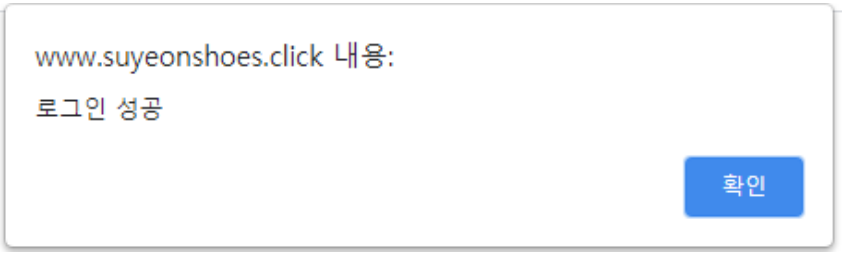
소비자가 회원가입을 진행하게되면 해당 소비자의 정보는 RDS에 저장되며, 로그인을 시도할 시 로그인 페이지에서 입력한 값은 데이터베이스에 저장된 값과 비교하여 로그인 성공 여부를 판단하게 된다.



[그림 6] 회원가입 페이지



[그림 7] 로그인 페이지



[그림 8] 로그인 성공

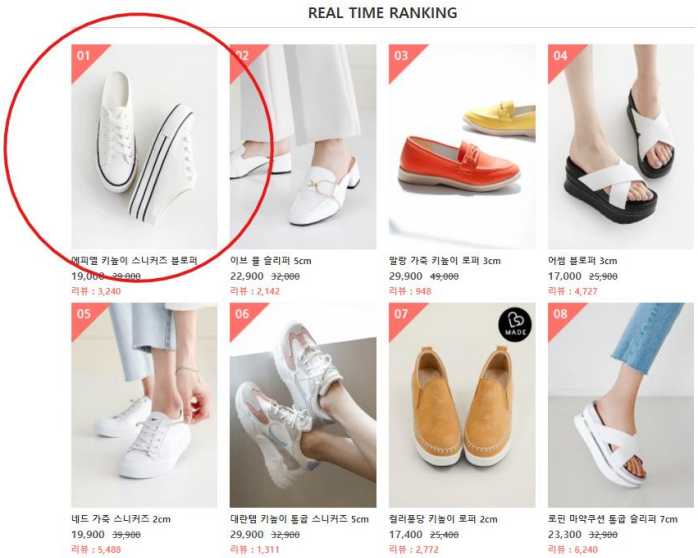
3.1.3. 스티키 세션을 사용한 데이터 유지

스티키 세션은 라운드 로빈으로 작동하는 Load balancer에 세션을 고정해서 사용자의 세션을 지속할수 있도록 도와주는 역할을 한다.

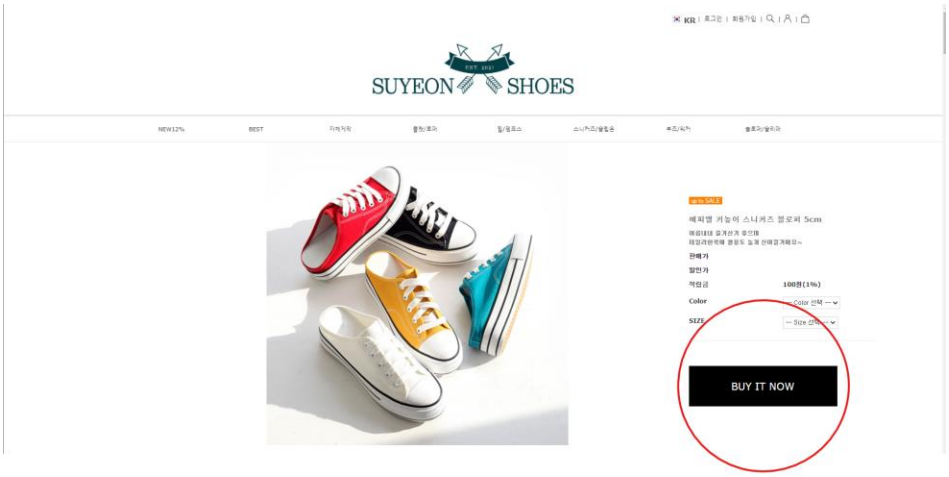
[그림 8] Sticky Session

3.1.4. 물품 구매

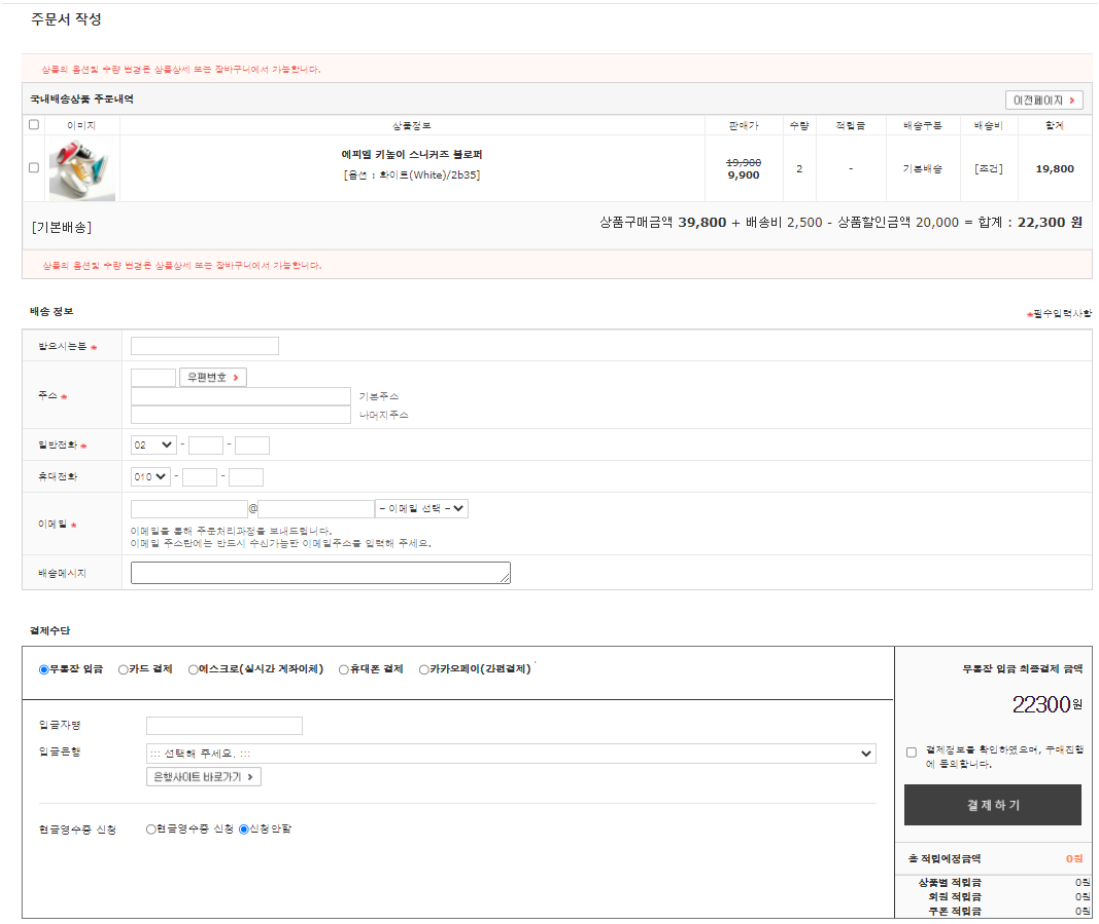
소비자가 상품 목록의 구매 버튼을 클릭하게 되면 상품 리스트를 볼 수 있다. 이때 쇼핑몰의 DB에서 해당상품의 값을 읽어와 소비자가 선택한 상품의 화면이 나와 구매까지 가능하다.



[그림 9] 상품 목록



[그림 10] 상품 구매 페이지

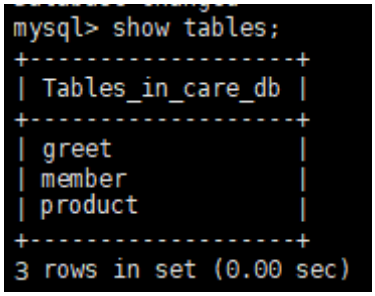


[그림 11] 상품 결제 페이지

4. 개발 및 인프라 서비스

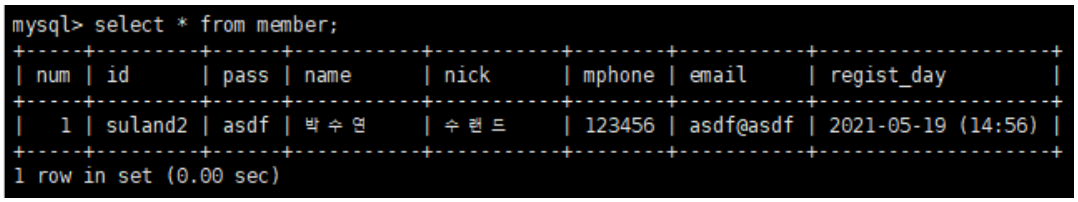
4.1. RDS

RDS는 관계형 데이터베이스를 구현하는 AWS 관리형 서비스이다. 본 프로젝트에서는 회원정보 테이블과 상품목록 테이블을 생성하여 사용했다.



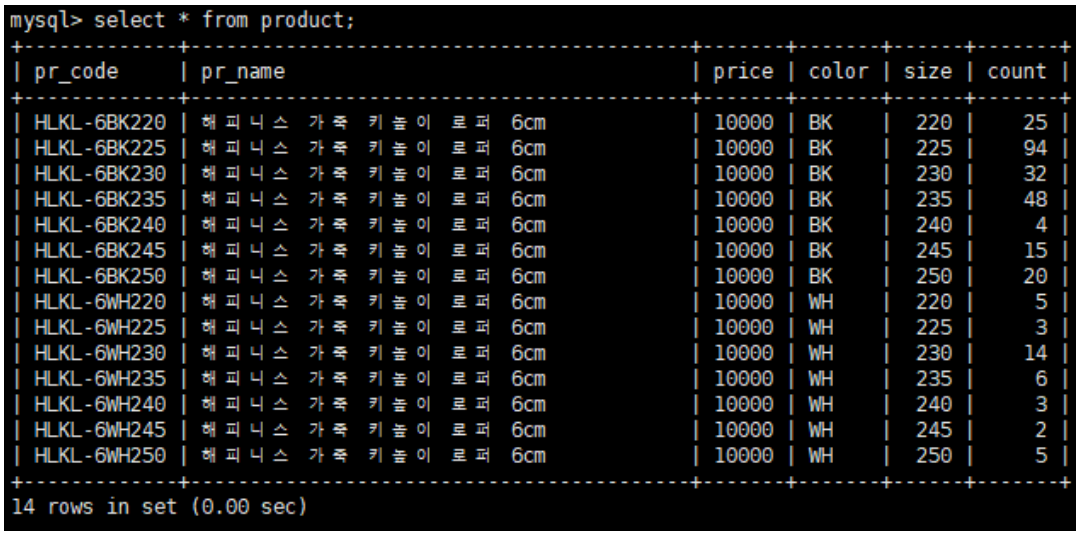
```
mysql> show tables;
+-------------------+
| Tables_in_care_db |
+-------------------+
| greet             |
| member            |
| product           |
+-------------------+
3 rows in set (0.00 sec)
```

[그림 12] Care_DB안 테이블 목록



```
mysql> select * from member;
+-----+---------+------+-----------+-----------+--------+-----------+---------------------+
| num | id      | pass | name      | nick      | mphone | email     | regist_day          |
+-----+---------+------+-----------+-----------+--------+-----------+---------------------+
|   1 | suland2 | asdf | 박수연    | 수랜드    | 123456 | asdf@asdf | 2021-05-19 (14:56) |
+-----+---------+------+-----------+-----------+--------+-----------+---------------------+
1 row in set (0.00 sec)
```

[그림 13] 회원정보 테이블

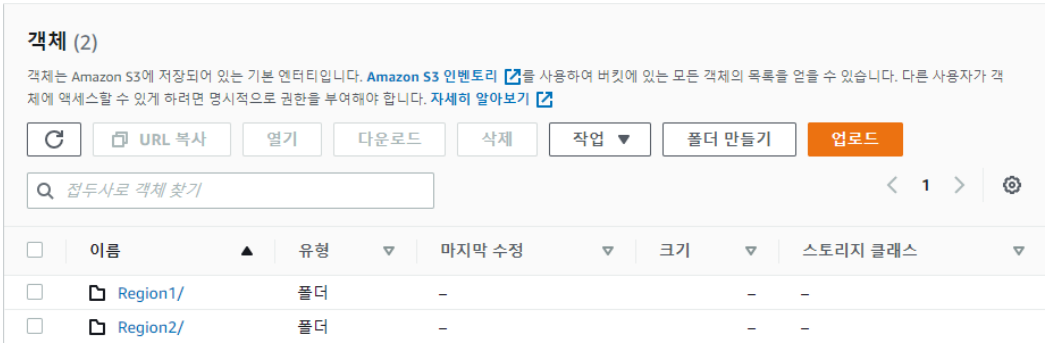


```
mysql> select * from product;
+-------------+-----------------------------------------+-------+-------+------+-------+
| pr_code     | pr_name                                 | price | color | size | count |
+-------------+-----------------------------------------+-------+-------+------+-------+
| HLKL-6BK220 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  220 |    25 |
| HLKL-6BK225 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  225 |    94 |
| HLKL-6BK230 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  230 |    32 |
| HLKL-6BK235 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  235 |    48 |
| HLKL-6BK240 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  240 |     4 |
| HLKL-6BK245 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  245 |    15 |
| HLKL-6BK250 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  250 |    20 |
| HLKL-6WH220 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  220 |     5 |
| HLKL-6WH225 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  225 |     3 |
| HLKL-6WH230 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  230 |    14 |
| HLKL-6WH235 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  235 |     6 |
| HLKL-6WH240 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  240 |     3 |
| HLKL-6WH245 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  245 |     2 |
| HLKL-6WH250 | 헤피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  250 |     5 |
+-------------+-----------------------------------------+-------+-------+------+-------+
14 rows in set (0.00 sec)
```

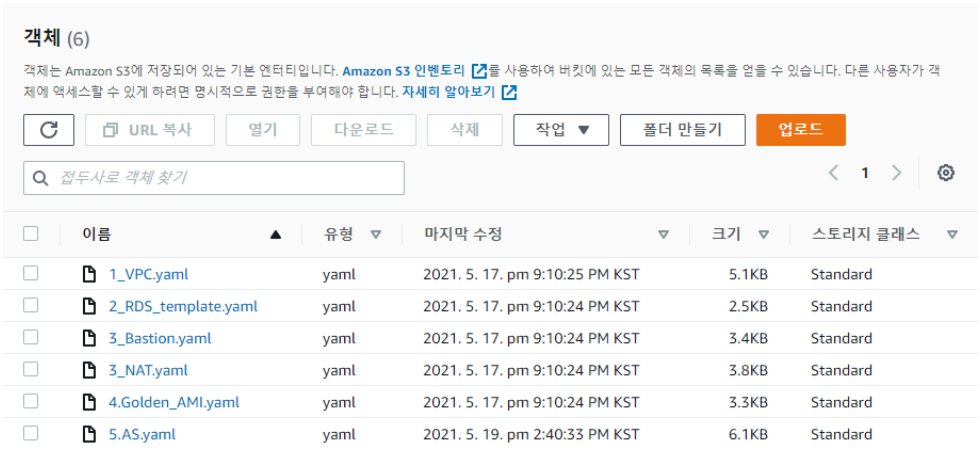
[그림 14] 상품목록 테이블

4.2. S3

S3는 파일을 저장하는 저장소 서비스이다. 본 프로젝트에서는 S3를 Cloudformation Template 파일을 업로드 하는 대상으로 사용하였다. 재해 복구할 때 필요한 리전의 Cloudformation Template이 저장된 S3 버킷의 주소를 활용하였다. 또한 데이터베이스 속 데이터들을 엑셀파일로 정리하였고, 이를 S3에 넣어 관리하는 방식으로 진행하였다.



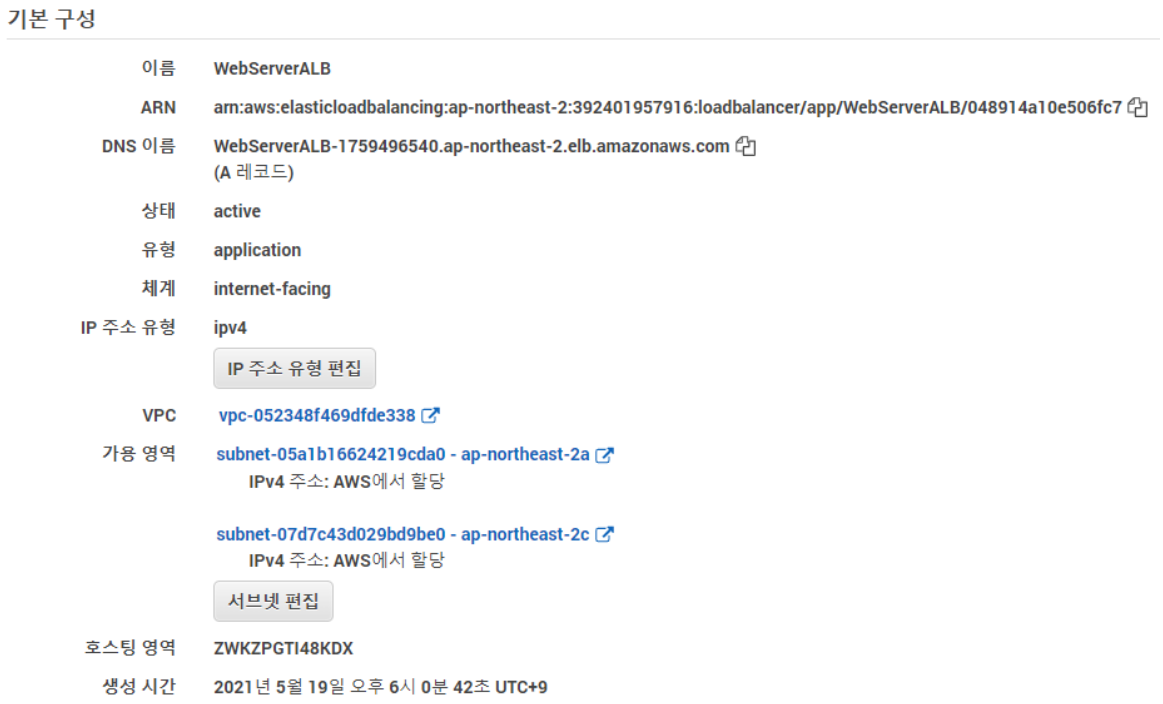
[그림 15] S3 Bucket



[그림 16] S3 Bucket/Region 1

4.3. Load Balancer

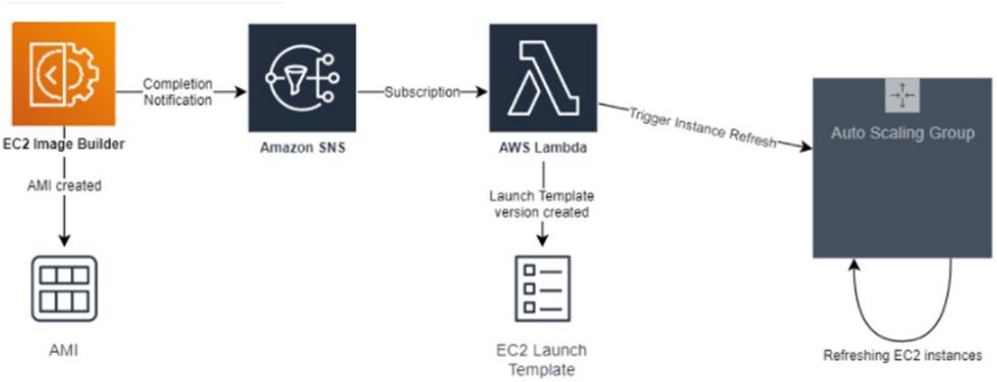
Load Balancers는 트래픽을 분산시켜 부하를 완화하는 역할을 한다. 그 중에서도 ALB를 사용하였고 웹 서버는 80(Http)포트를 통해 트래픽을 받으므로 Load Balancer의 Listener ID는 HTTP:80으로 구성하였다.



[그림 17] Load Balancers

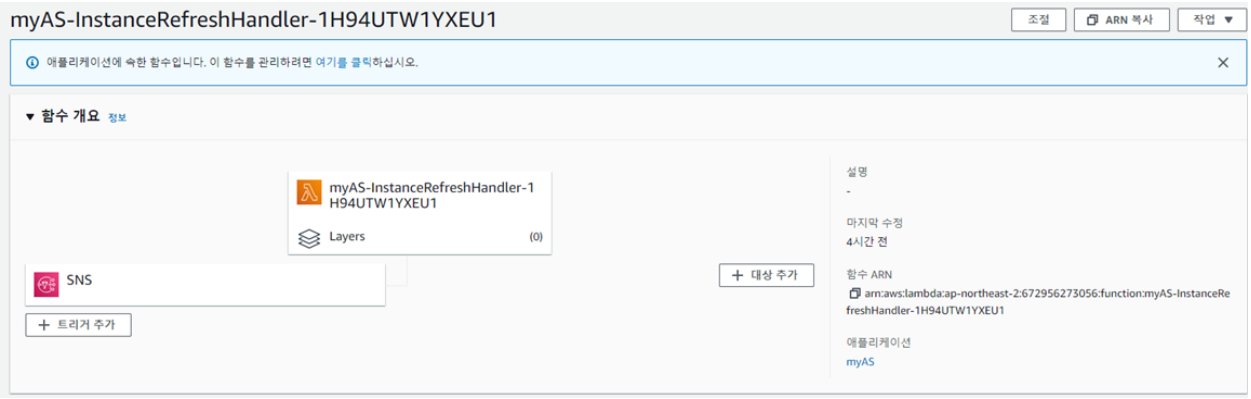
4.4. EC2 Image Builder

EC2 Image Builder를 통해 이미지를 관리 및 배포한다. Image Builder를 사용하면 이미지를 업데이트하기 위한 수동 단계가 필요하지 않으며, 자체 자동화 파이프라인을 구축하지 않아도 된다는 장점이 있다. 소프트웨어 업데이트가 제공되면 Image Builder는 자동으로 새 이미지를 생성하고 테스트를 실행한 후 지정한 리전으로 배포한다.



[그림 18] EC2 Image Builder, SNS , Lambda

4.4.1. SNS



[그림 19] SNS , Lambda

EC2 Image Builder의 이미지 배포가 끝나게 되면 SNS를 통하여 Lambda함수가 작동되도록 한다.

4.4.2. Lambda

Lambda는 현재 Auto Scaling의 Launch Template의 이미지를 EC2 Image Builder를 통해 배포된 새로운 이미지로 구성한다. 그 이후 Auto Scaling Group을 Refresh하여 구 버전의 인스턴스를 Scale In 하고, 새로운 버전의 인스턴스를

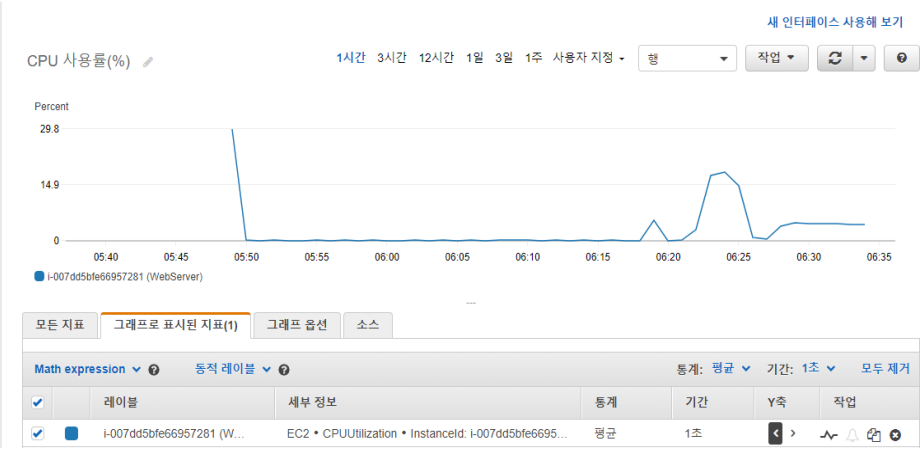
Scale Out하여 롤링배포를 한다. 사용자는 업데이트되는 서비스를 서비스 중단 없이 사용할 수 있다.

4.5. EC2 Auto Scaling

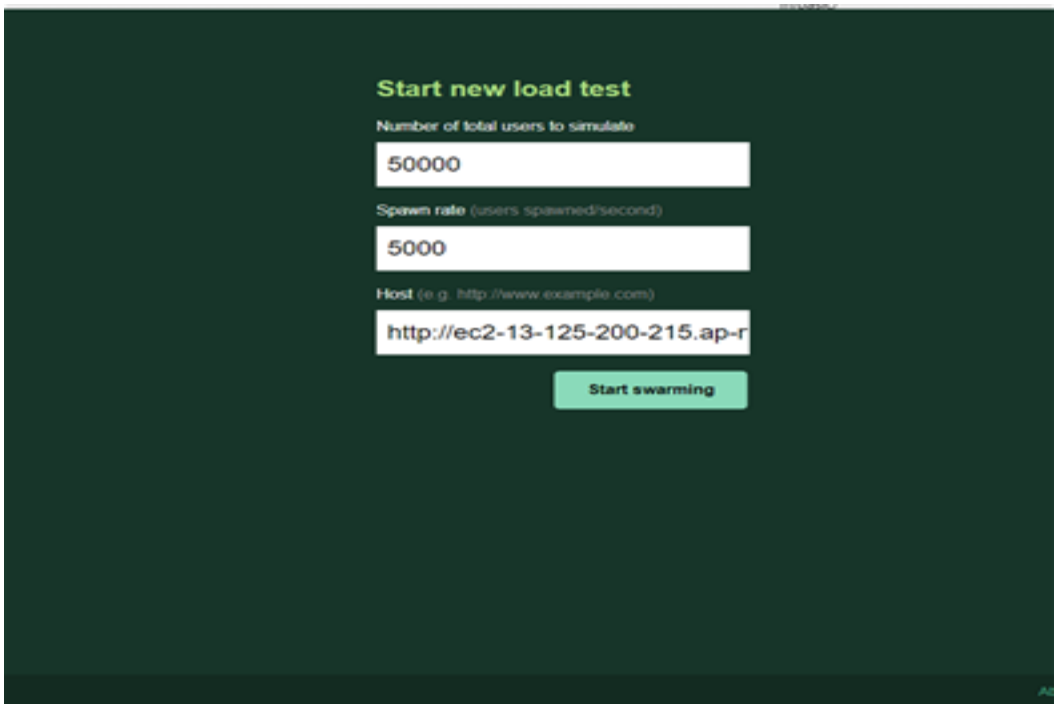
4.5.1. Locust를 이용한 부하 테스트

테스트를 위해 Locust라고 하는 오픈소스 트래픽 부하테스트 툴을 사용했다.

시뮬레이션의 총 사용자는 50,000명으로 가정하고, 초당 사용자 생성 속도는 1000명, 테스트 대상 사이트는 쇼핑몰 주소로 설정했다.



[그림 20] CPU Utilization



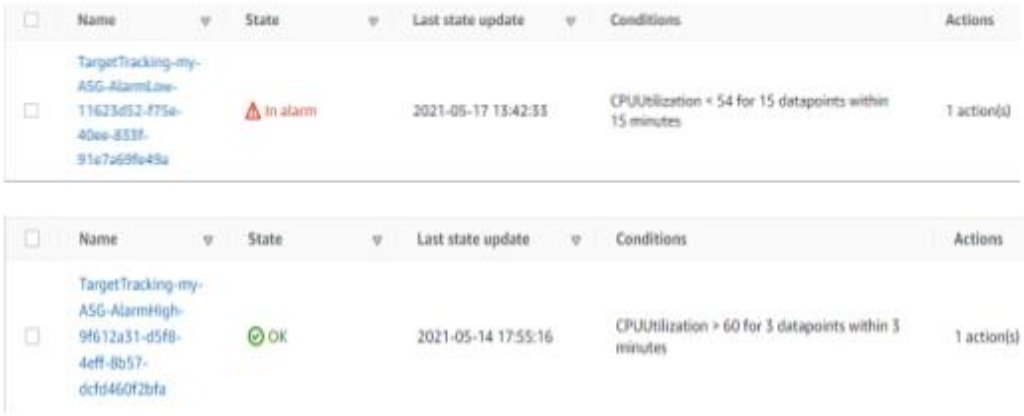
[그림 21] Locust 부하테스트 설정

[그림 22] Locust 부하테스트 실행

4.5.2. CloudWatch

CloudWatch는 애플리케이션을 모니터링하고, 시스템 전반의 성능 변경 사항에

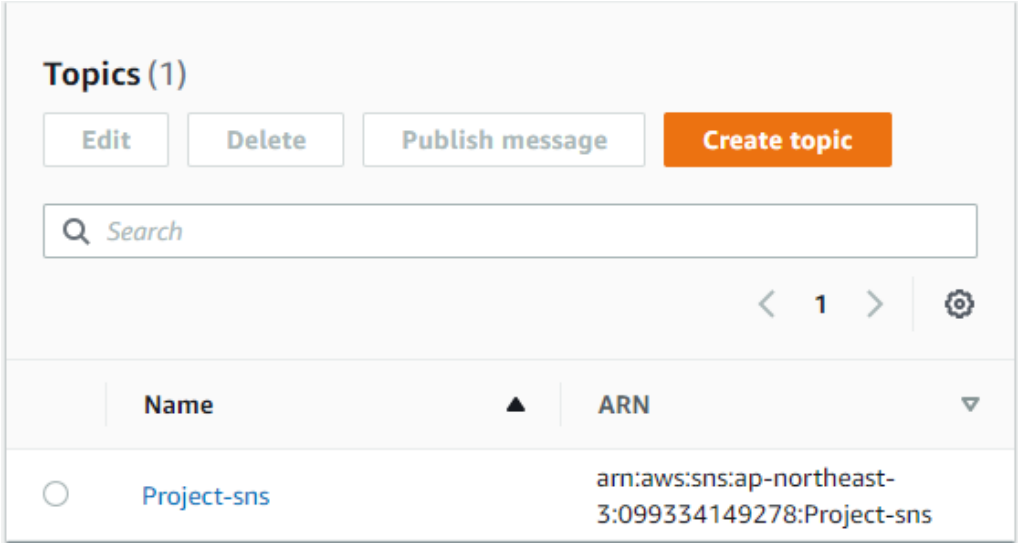
대응하며, 리소스 사용률을 최적화하고, 운영 상태의 각종 지표에 기반한 알람 서비스이다. 본 프로젝트에서는 CPU 사용률 60%의 정의된 임계값에 대하여 부하 이상이 발생할 경우에 CloudWatch 경보가 발생된다.



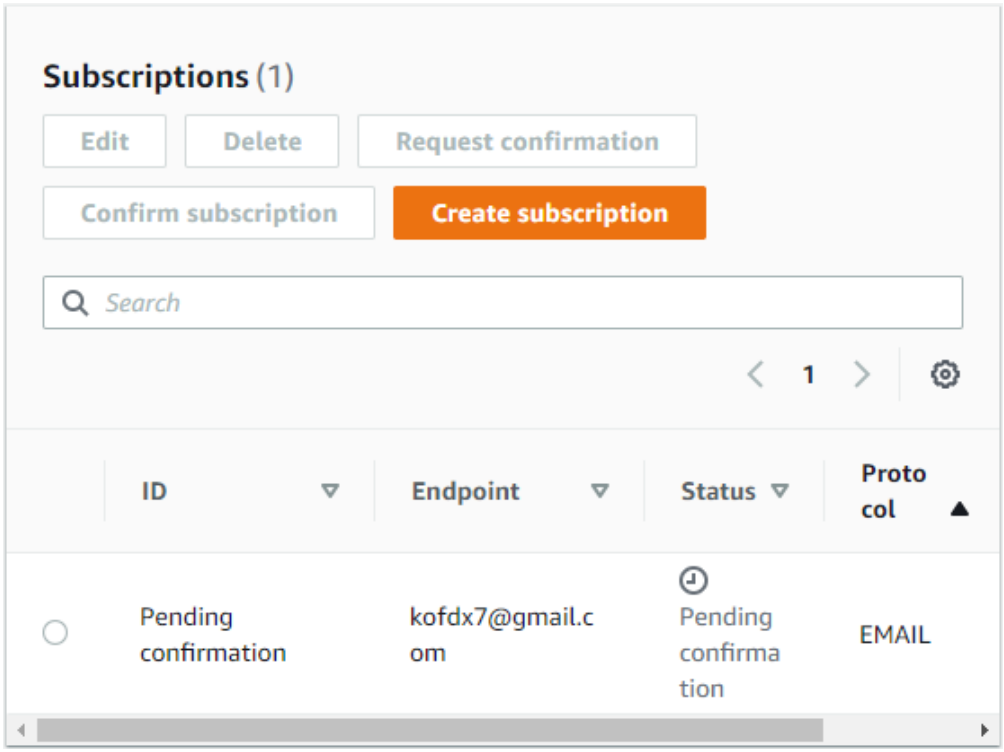
[그림 23] CloudWatch Alarm

4.5.3. SNS 알림전송

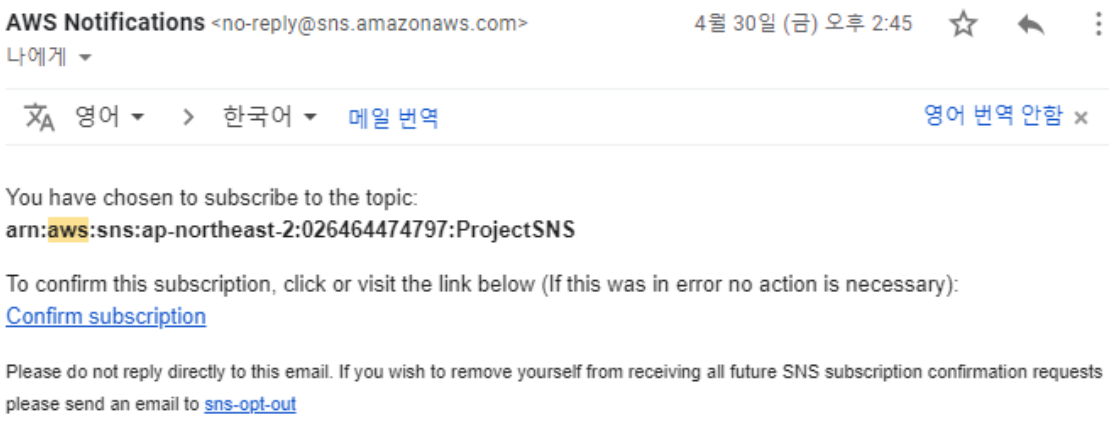
AWS의 SNS(Simple Notification Service)는 푸시 알림 서비스이며 애플리케이션 간(A2A) 및 애플리케이션과 사용자 간(A2P) 통신 모두를 위한 완전 관리형 메시징 서비스이다. 처음에는 Subscription된 이메일로 Confirm 메일을 보내며 그 이후에는 등록된 이메일로 Topic에 대한 메일을 보내게 된다.



[그림 24] SNS Topic



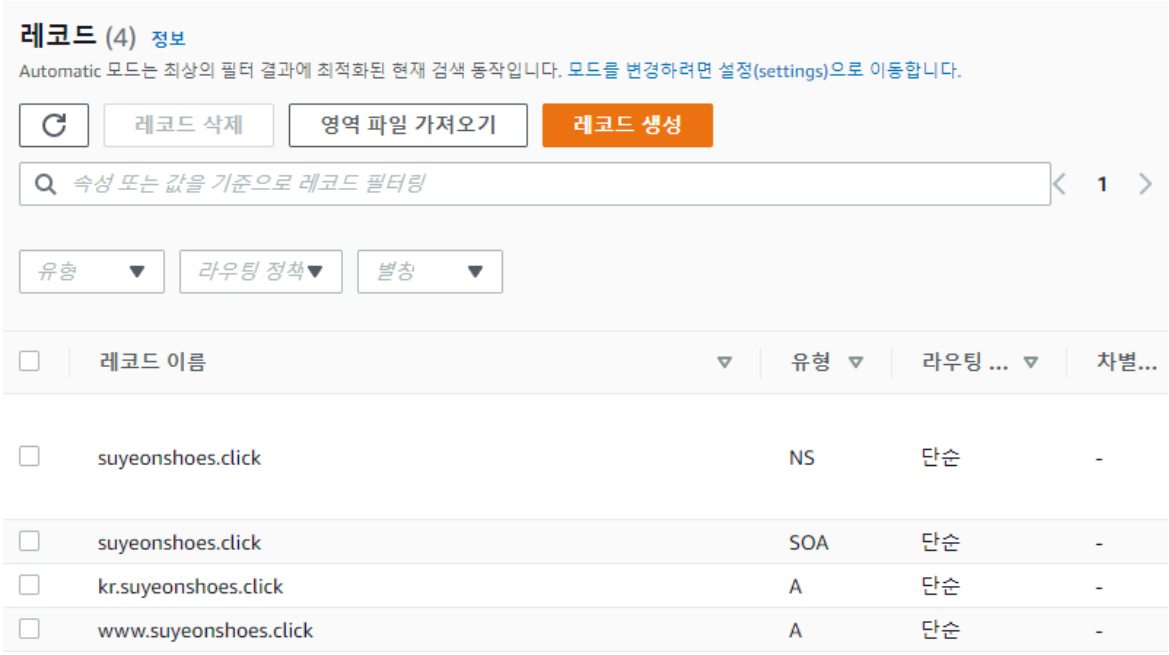
[그림 25] SNS Subscriptions



[그림 26] Confirmation Email for SNS

4.6. Route 53

Route 53은 도메인 생성 및 한국 도메인과 일본 도메인 구분을 위한 레코드 관리를 위해 사용하였다.



[그림 27] Route 53 레코드

5. CloudFormation

CloudFormation은 코드를 통해 Amazon Web Services 리소스를 모델링하고 배포한다, 관리 시간을 줄이고 AWS에서 실행되는 애플리케이션에 집중할 수 있도록 해주는 서비스다. 본 프로젝트에서는 모든 개발 환경 구성, 재해복구(DR)에 대하여 템플릿을 구성하였다.

5.1. Seoul & Osaka Region

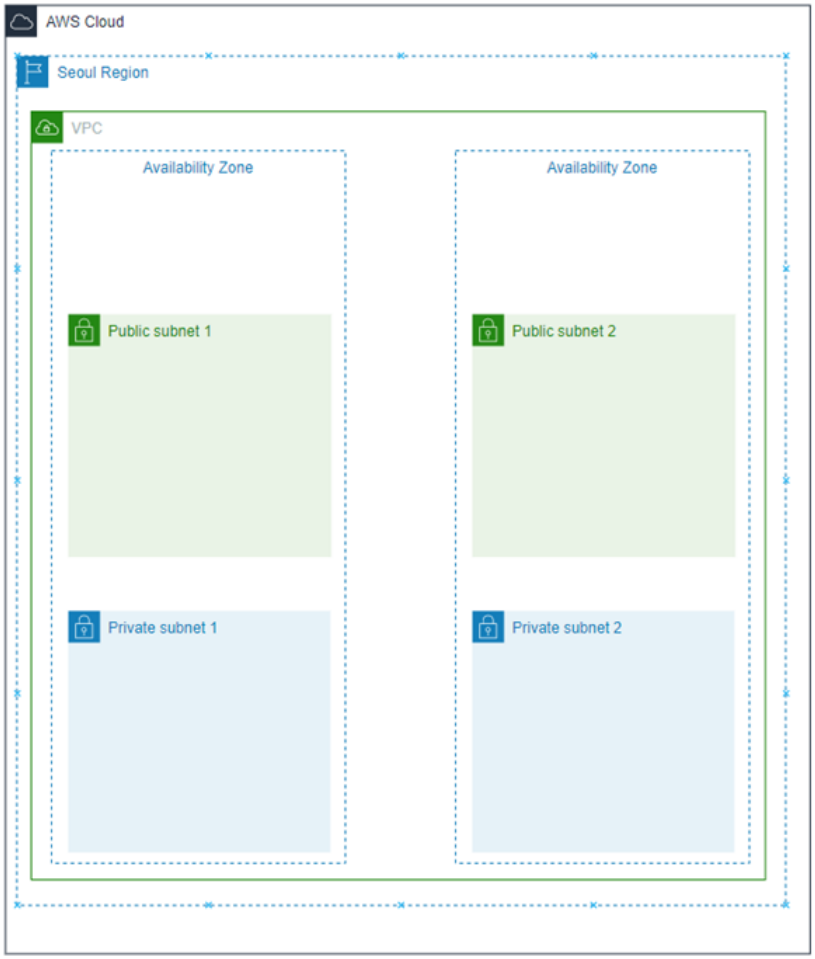
Stack name	Status
myRoute53	CREATE_COMPLETE
myAS	CREATE_COMPLETE
myGolden	CREATE_COMPLETE
myNAT	CREATE_COMPLETE
myBastion	CREATE_COMPLETE
myRDS	CREATE_COMPLETE
myVPC	UPDATE_COMPLETE

[그림 28] CloudFormation Template

5.1.1. VPC

논리적으로 격리된 가상 네트워크로서, 가상 사설 클라우드를 사용자에게 제공하는 상업용 클라우드 컴퓨팅 서비스다. VPC를 AWS 클라우드에서 논리적으로 격리된 공간을 프로비저닝하여 고객이 정의하는 가상 네트워크에서 AWS 리소스를 시작할 수 있다.

가용영역 2개(a,c)를 생성 후 해당 영역에 Public Subnet, Private Subnet을 생성하였다.



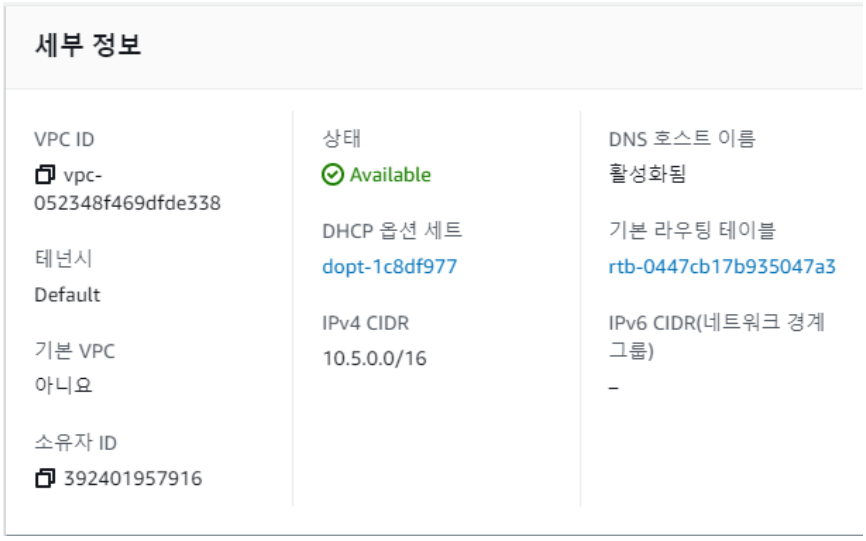
[그림 29] Seoul Region VPC 구성

가용영역 A

- Public Subnet1 : 10.5.10.0/24, Private Subnet1 : 10.5.20.0/24

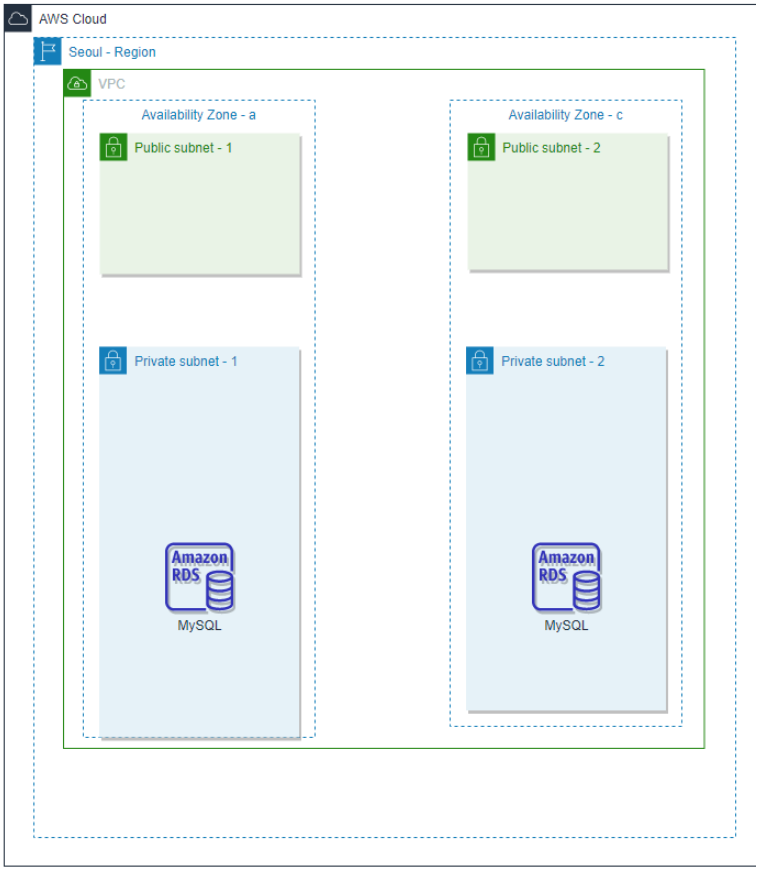
가용영역 C

- Public Subnet2 : 10.5.30.0/24, Private Subnet2 : 10.5.40.0/24

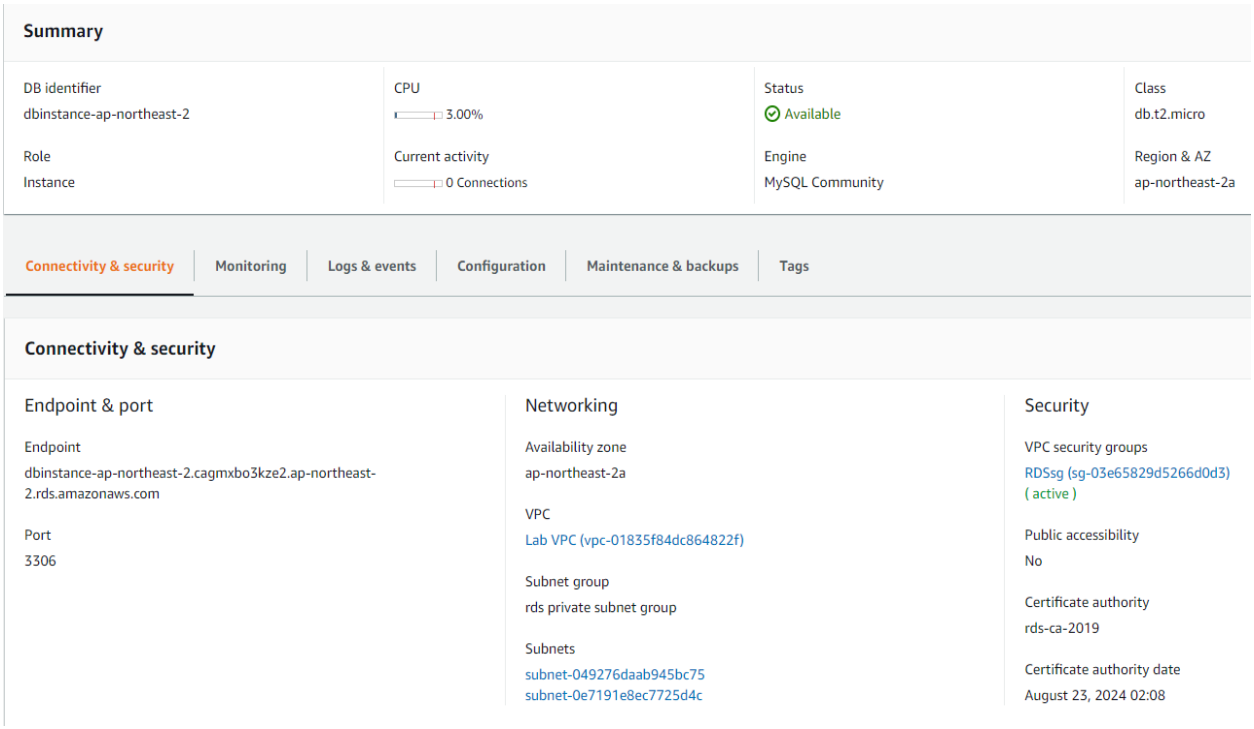


[그림 30] VPC 구성 (부록 1 참고)

5.1.2. RDS



[그림 31] Seoul Region RDS 구성

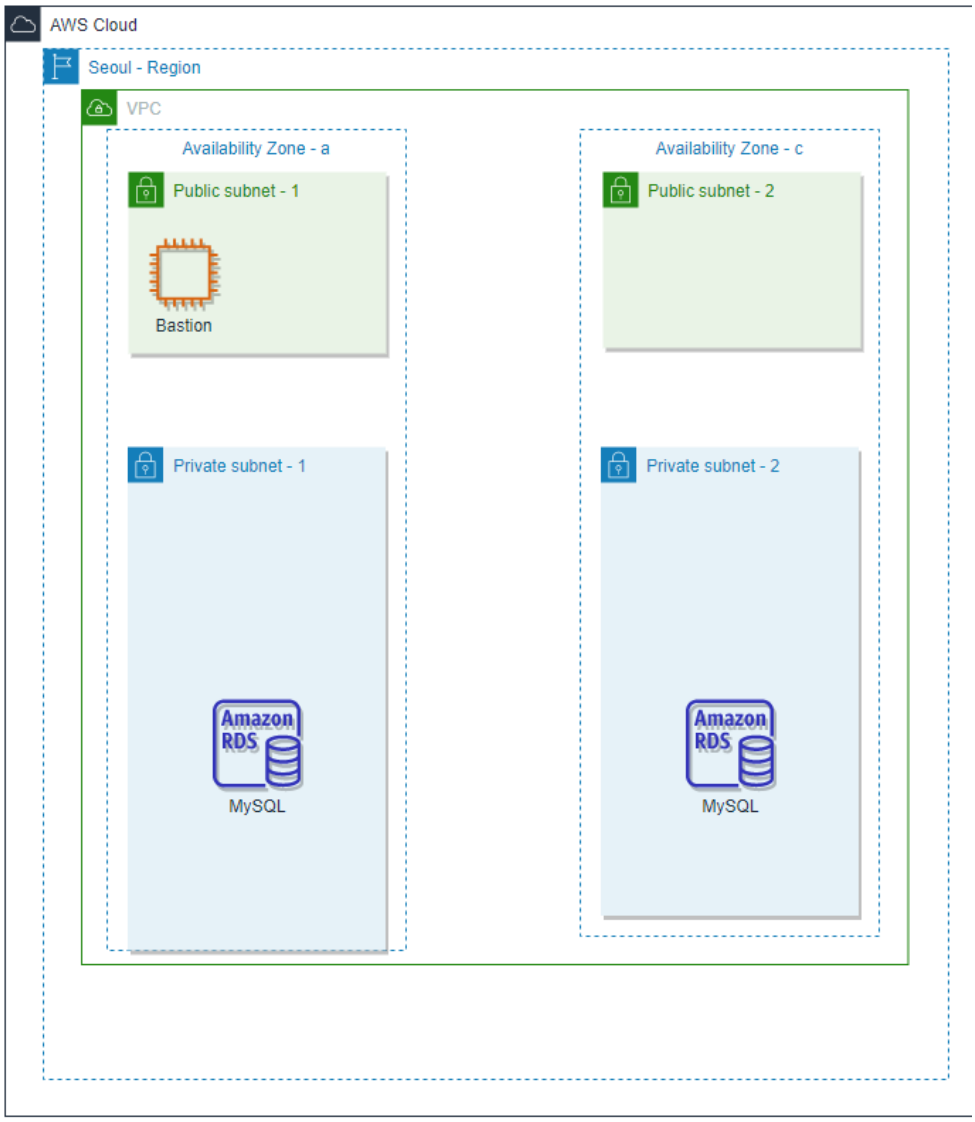


[그림 32] Seoul Region RDS 생성 (부록 2 참고)

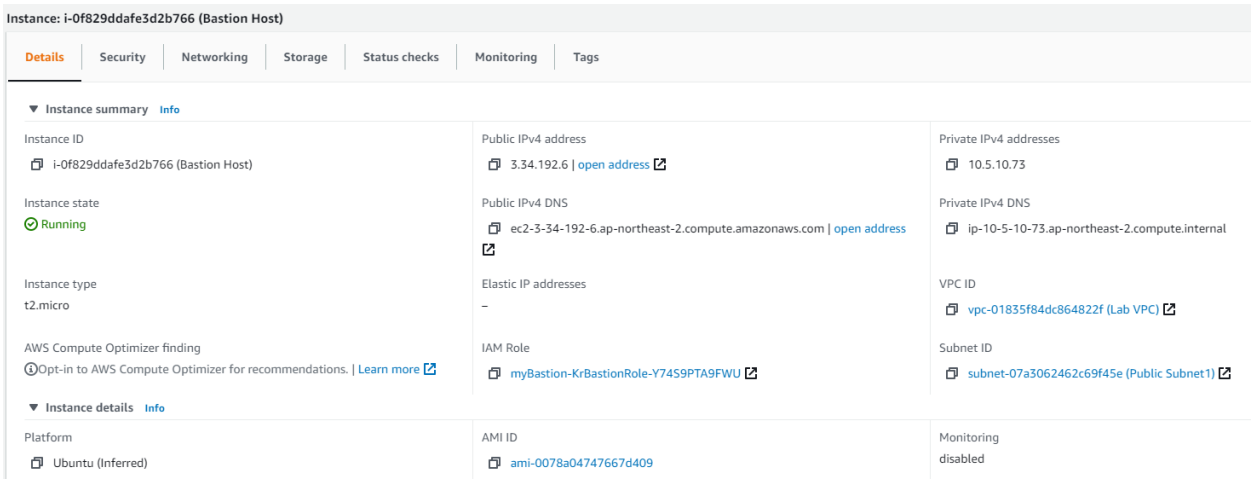
```
mysql> select * from product;
+------------+------------------------------------------+-------+-------+------+-------+
| pr_code    | pr_name                                  | price | color | size | count |
+------------+------------------------------------------+-------+-------+------+-------+
| HLKL-6BK220 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  220 |    25 |
| HLKL-6BK225 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  225 |    94 |
| HLKL-6BK230 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  230 |    32 |
| HLKL-6BK235 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  235 |    48 |
| HLKL-6BK240 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  240 |     4 |
| HLKL-6BK245 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  245 |    15 |
| HLKL-6BK250 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | BK    |  250 |    20 |
| HLKL-6WH220 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  220 |     5 |
| HLKL-6WH225 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  225 |     3 |
| HLKL-6WH230 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  230 |    14 |
| HLKL-6WH235 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  235 |     6 |
| HLKL-6WH240 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  240 |     3 |
| HLKL-6WH245 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  245 |     2 |
| HLKL-6WH250 | 해피니스 가죽 키높이 로퍼 6cm           | 10000 | WH    |  250 |     5 |
+------------+------------------------------------------+-------+-------+------+-------+
14 rows in set (0.00 sec)
```

[그림 33] DB Table 생성 및 확인

5.1.3. Bastion

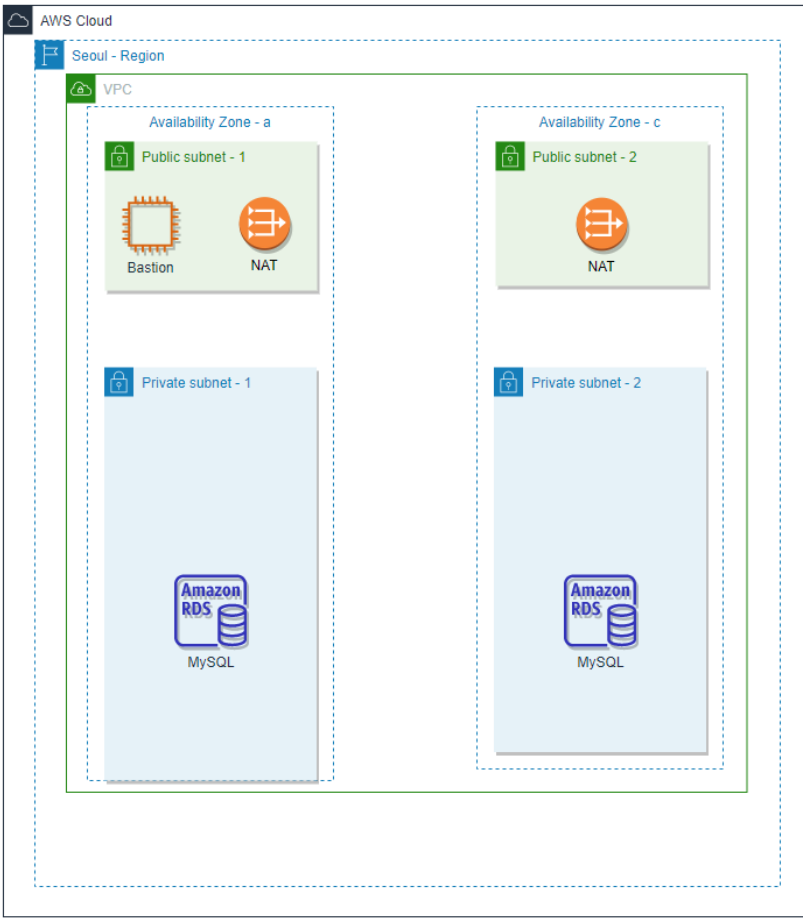


[그림 34] Seoul Region Bastion Host 구성

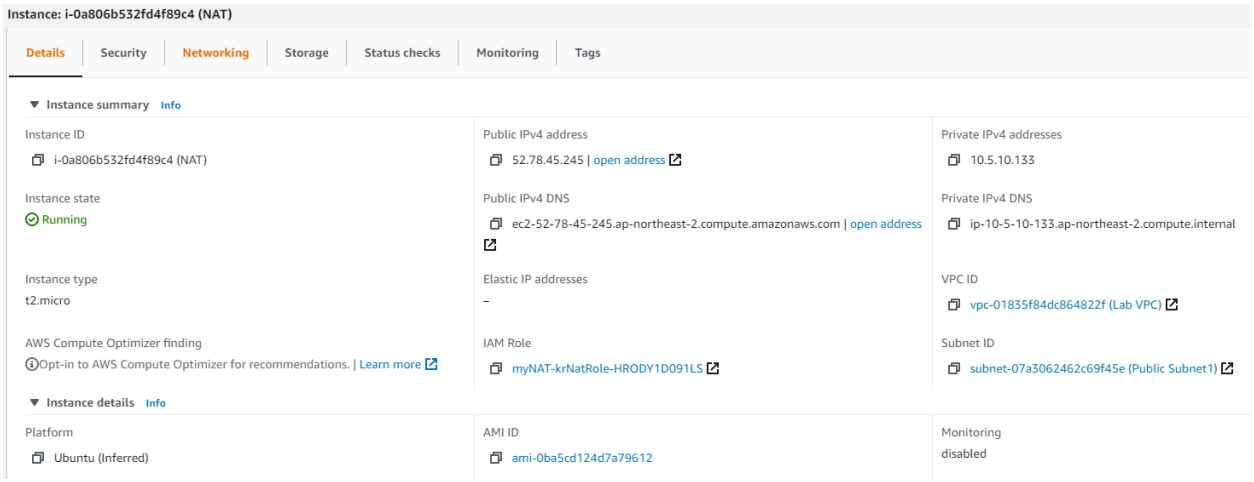


[그림 35] Bastion Host 구성 (부록 3 참고)

5.1.4. NAT

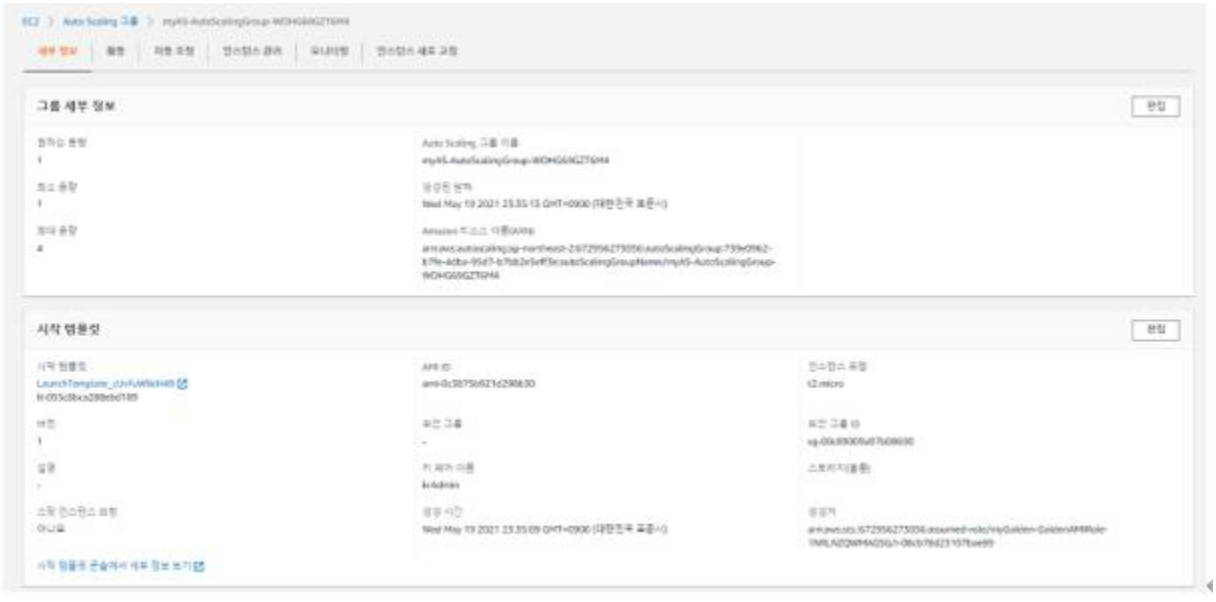


[그림 36] Seoul Region Nat 구성

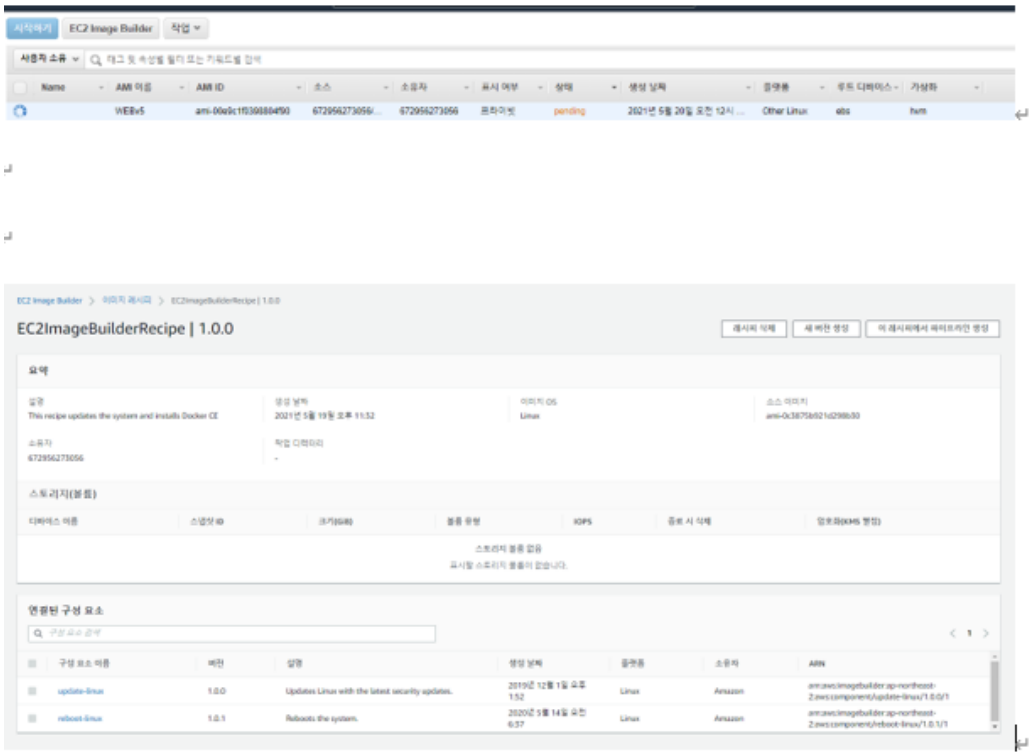


[그림 37] Seoul Region Nat 구성 (부록4 참고)

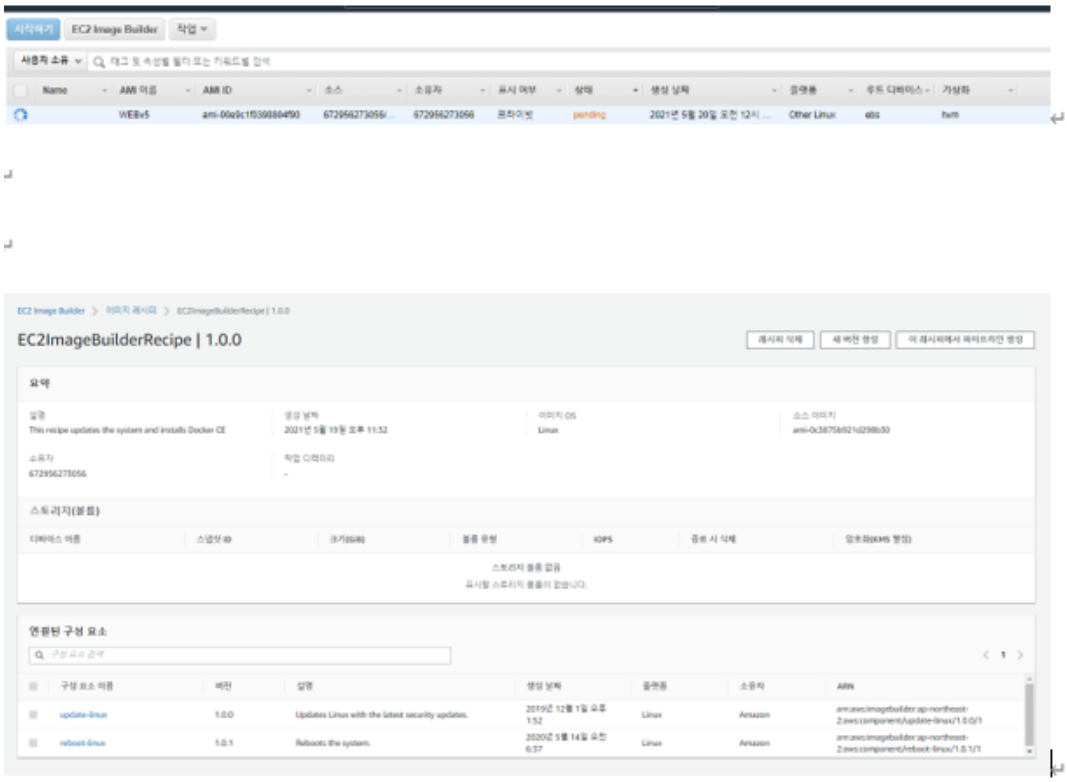
5.1.5. Golden AMI.yaml



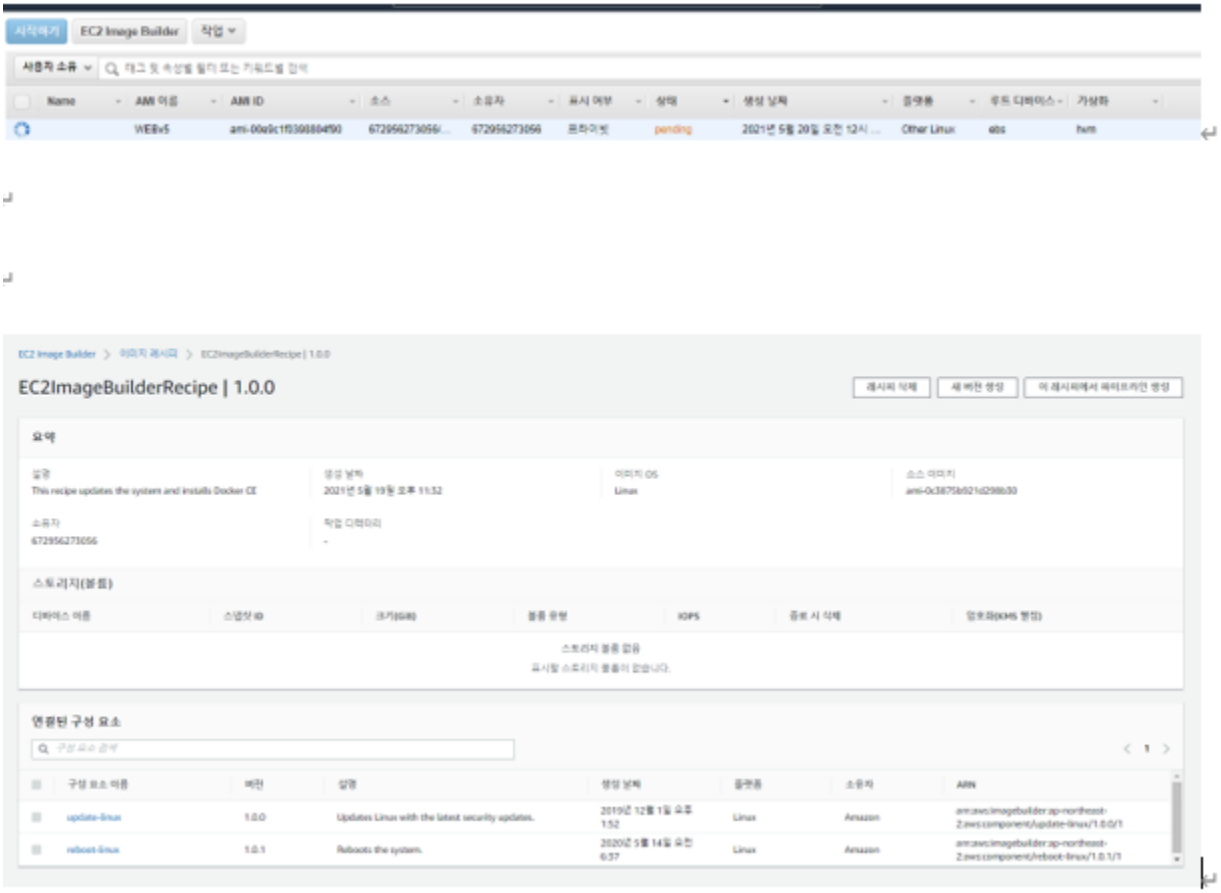
[그림 38] Auto Scaling group and Launch template 버전 설정



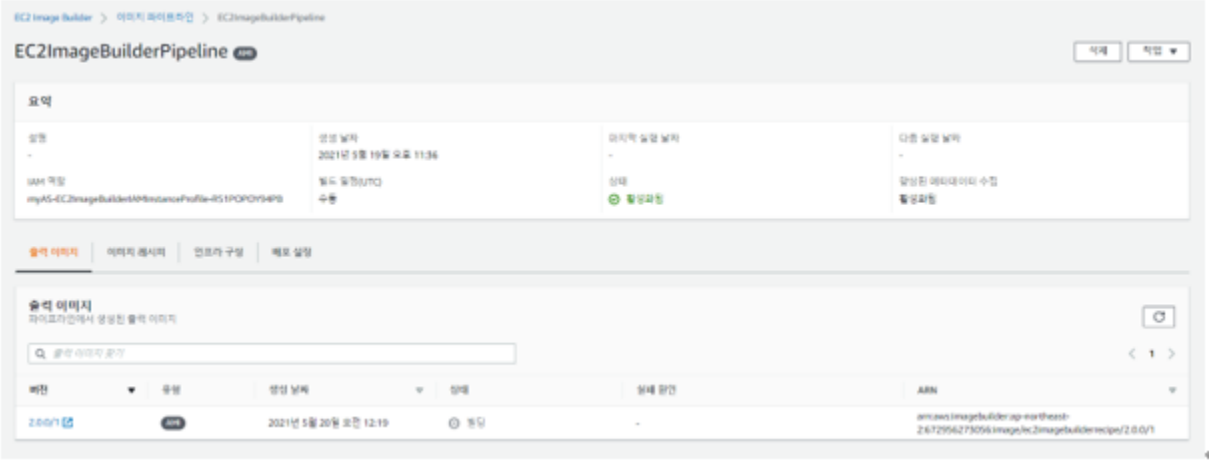
[그림 39] 새로운 버전의 이미지 생성



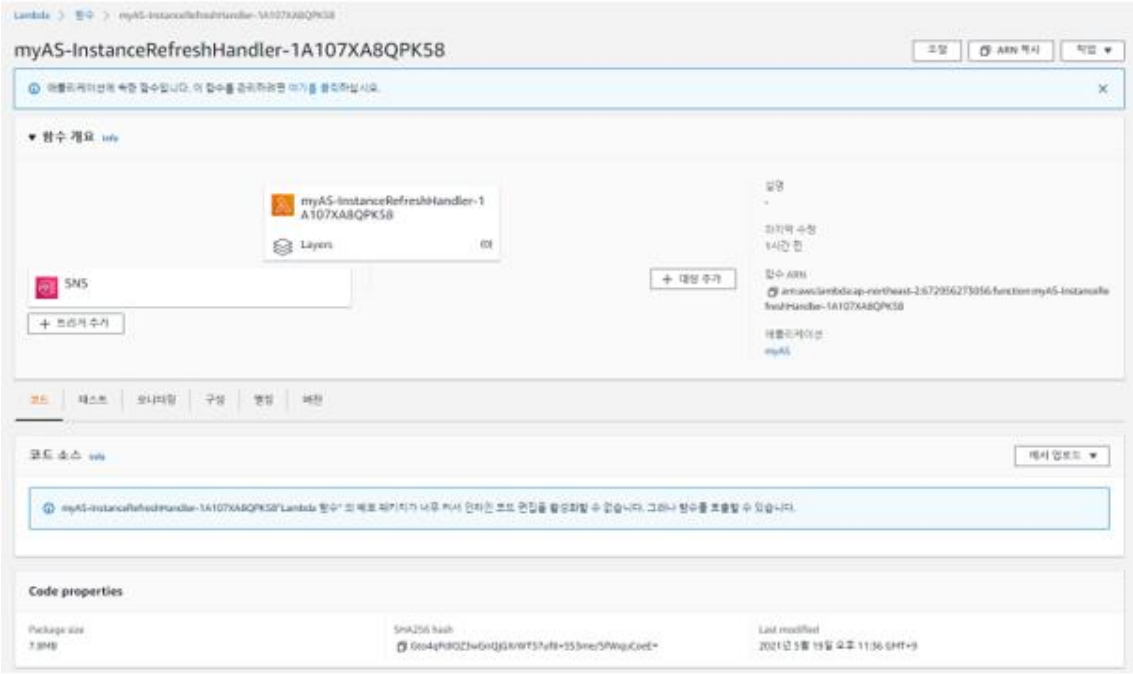
[그림 40] 새로운 이미지로 레시피 생성



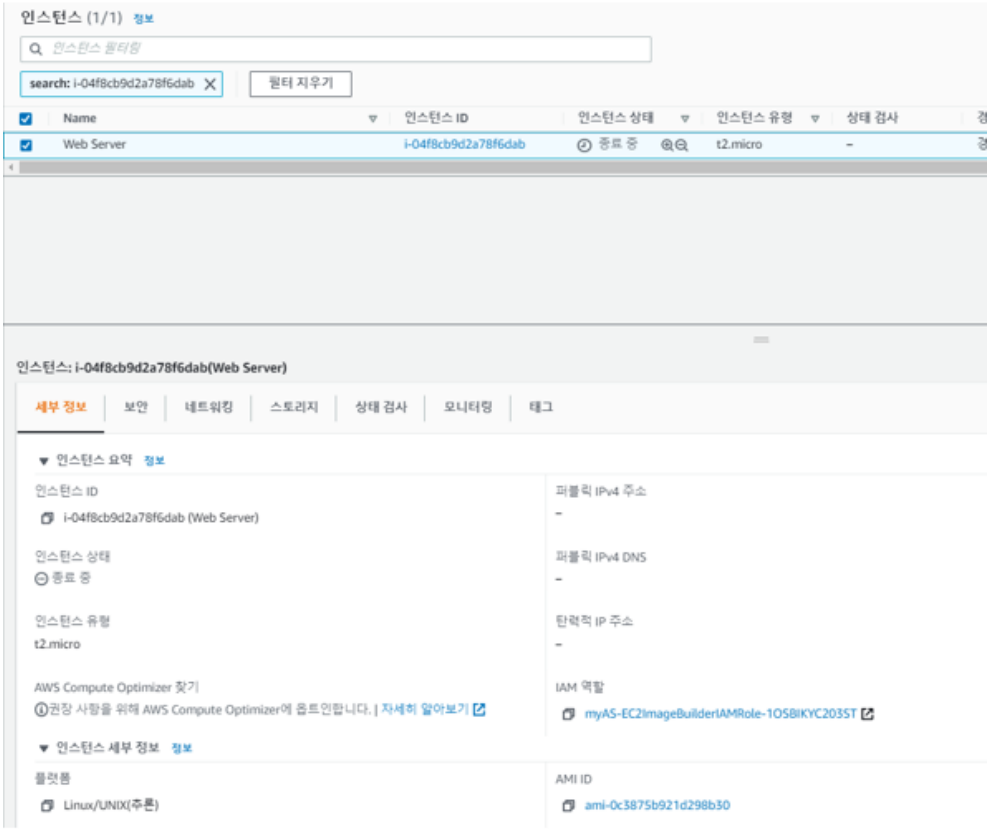
[그림 41] 파이프라인 편집



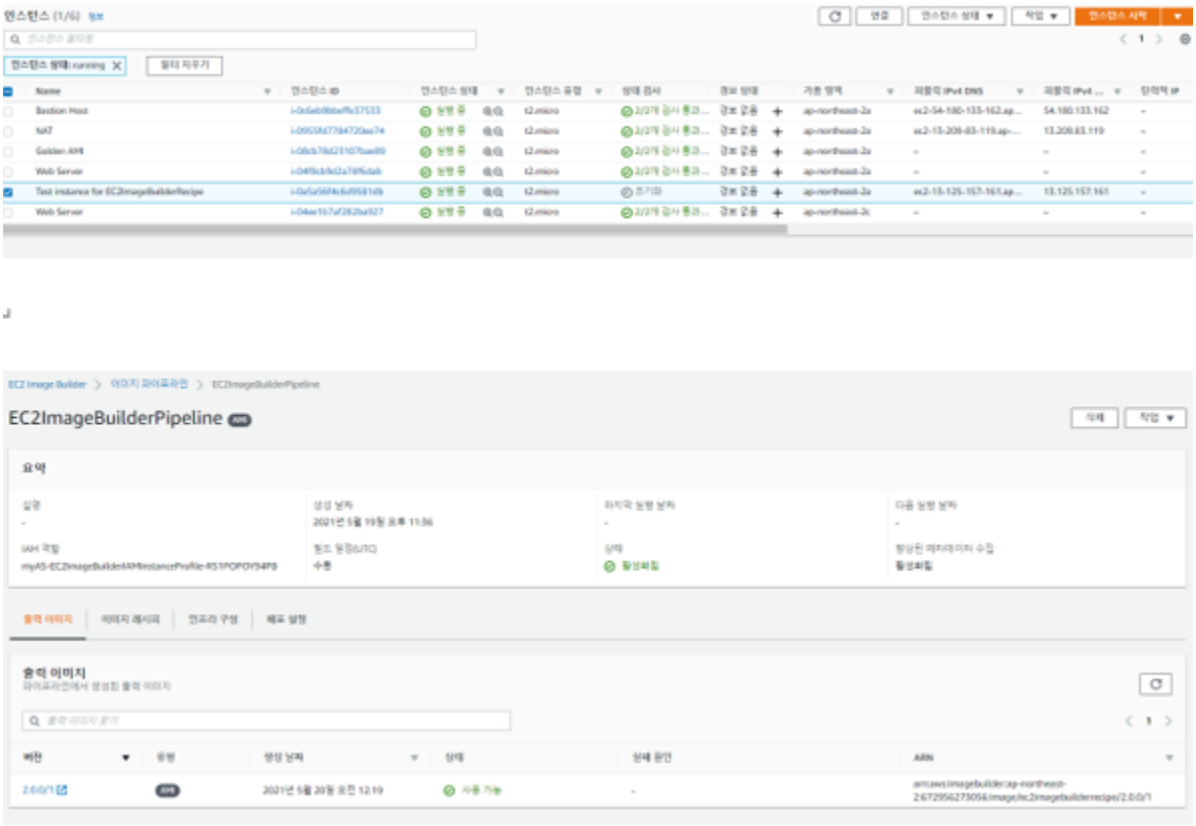
[그림 42] 파이프라인 실행



[그림 43] Auto Scaling Refresh

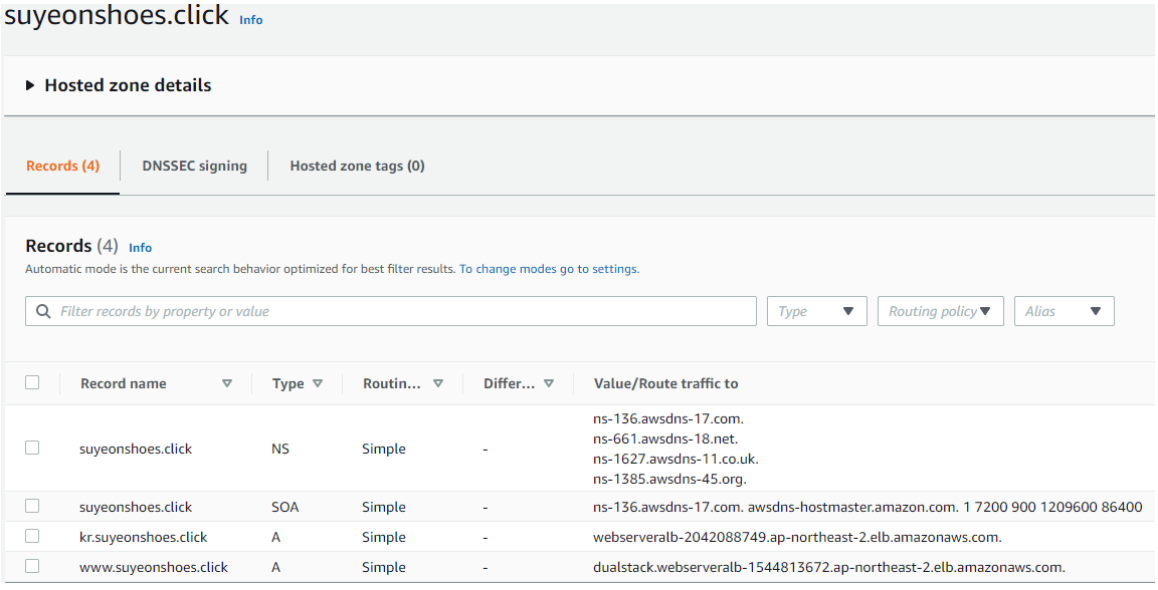


[그림 44] 구 버전의 인스턴스 종료



[그림 45] 새로운 버전의 이미지를 가진 인스턴스 시작

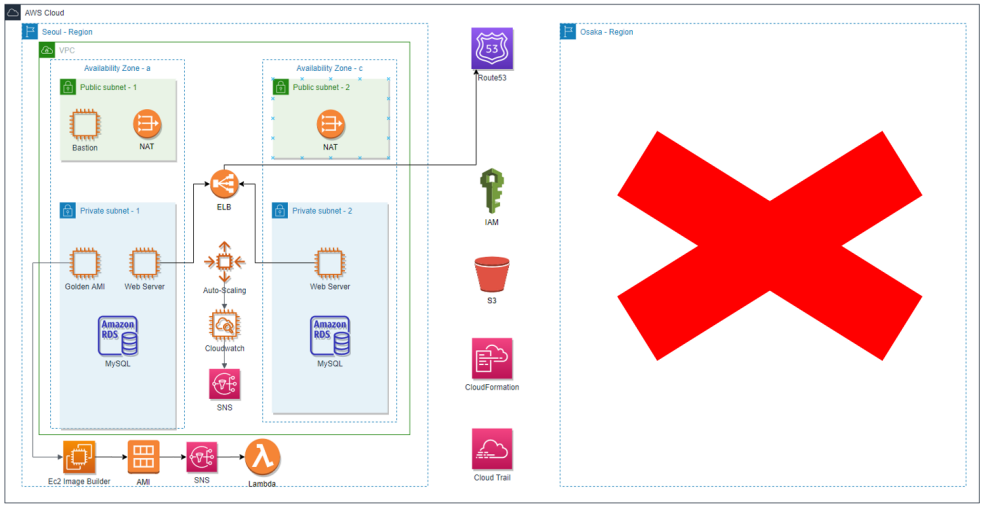
5.1.6. Route 53.yaml



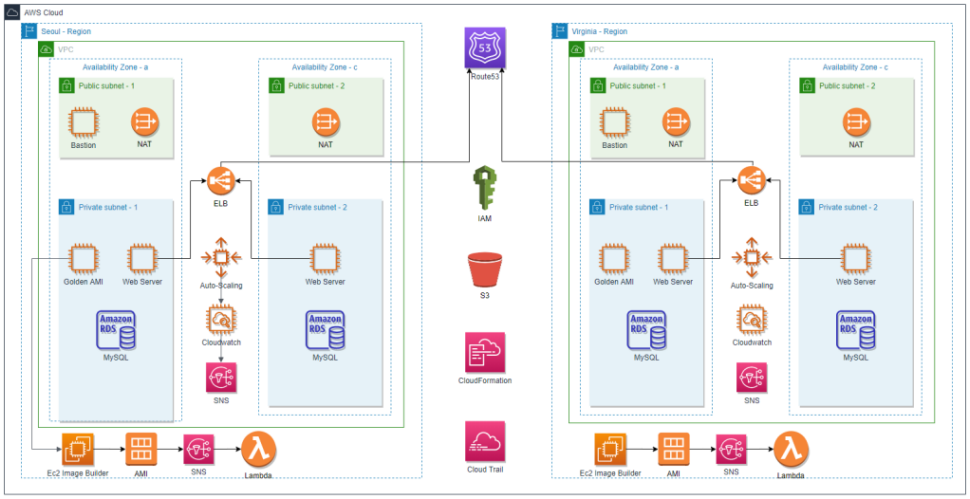
[그림 46] Route 53 설정 (부록 7 참고)

5.2. 템플릿 2 - 재해복구(DR) 구성

Seoul이나 Osaka Region 중 한 Region에 장애가 생겼을 때, 해당 Region이 아닌 Region에 모든 인프라와 서비스를 복구할 수 있도록 IAM 권한이 있는 관리자가 AWS CLI를 이용하여 재해복구를 진행한다.



[그림 47] Osaka Region 장애 발생



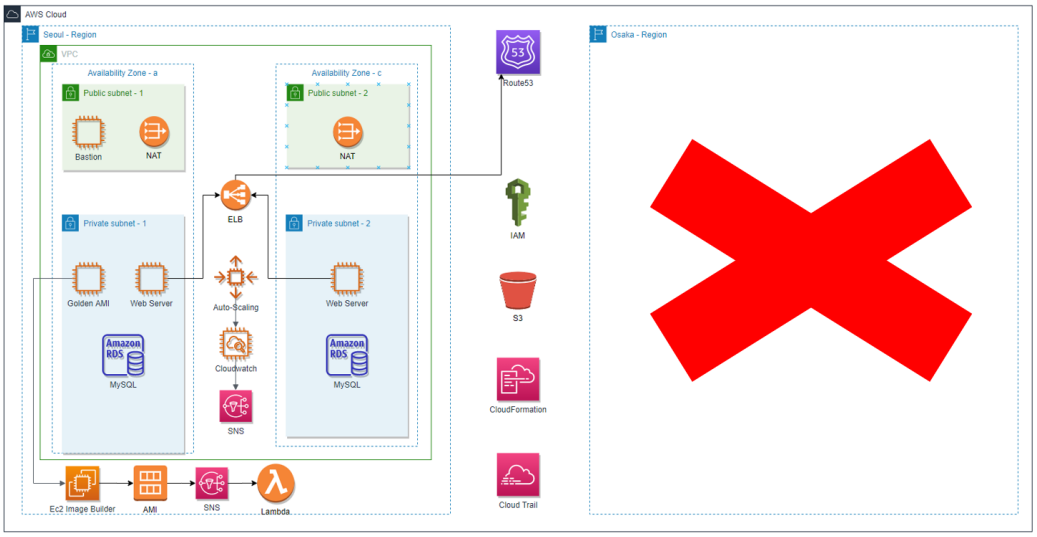
[그림 48] Virginia Region에 재해복구 완료

6. 재해복구(DR)

6.1. 재해복구(DR) - 시나리오

Seoul/Osaka Region 중 한 곳에 장애(화재, 홍수, 지진, 천재지변 등)가 발생하였을 경우 이에 대비하기 위한 기능이다. DR목표는 RTO(복구 목표 시간) 25분, RPO(복구 목표 지점) 0으로 설정했다.

장애를 감지하면 IAM 권한을 가진 관리자가 AWS CLI를 활용해 다른 Region에 재해복구를 시작한다. 해당 시점부터 Region에 필요한 모든 인프라와 서비스가 구축되는 시점까지 25분을 목표로 진행했고, 실제 21분정도 소요될 것으로 예상하고 있다. RPO는 0을 목표로 진행했다. RPO가 0이 아닐 경우, 장애가 발생했을 때 데이터 손실이 이루어진다. 본 프로젝트는 쇼핑몰을 대상으로 한 migration이기에 회원정보가 사라졌을 경우 소비자에게 불편함이, 구매정보가 사라졌을 경우 관리자에게 불편함이 생길 것이다. 어느 관점에서던 RPO를 최대한 줄이는 것이 좋으며, 본 프로젝트에서는 서로 다른 Region에 읽기 전용 복제본을 두어 실시간 backup을 통해 RPO가 0이 되는 것을 목표로 진행했다.



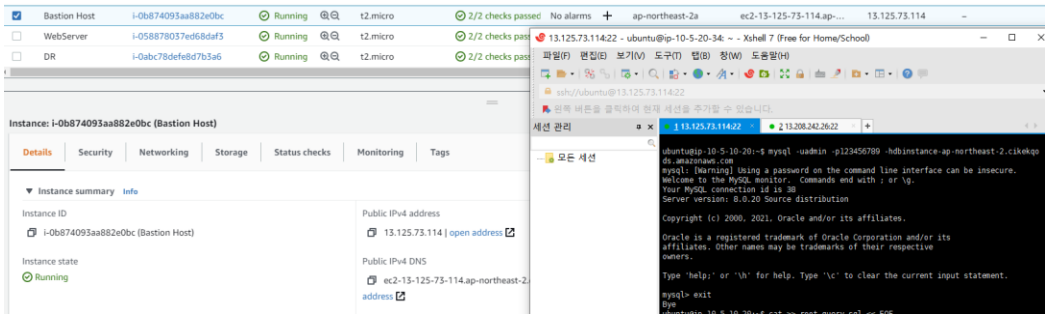
[그림 49] Osaka Region에 장애 발생

6.2. 재해복구(DR) - 준비단계

해당 단계를 위해 준비해야할 것은 2가지 AWS CLI환경, IAM DR user다. 관리자가 재해복구를 할 수 있도록 DR user IAM권한이 우선적으로 필요하다. 권한이 주어진 후 재해복구를 실행할 AWS CLI환경을 준비한다. 두 가지가 준비되면 재해복구를 위한 준비는 마무리가 된다.

6.3. 재해복구(DR) - 복구단계

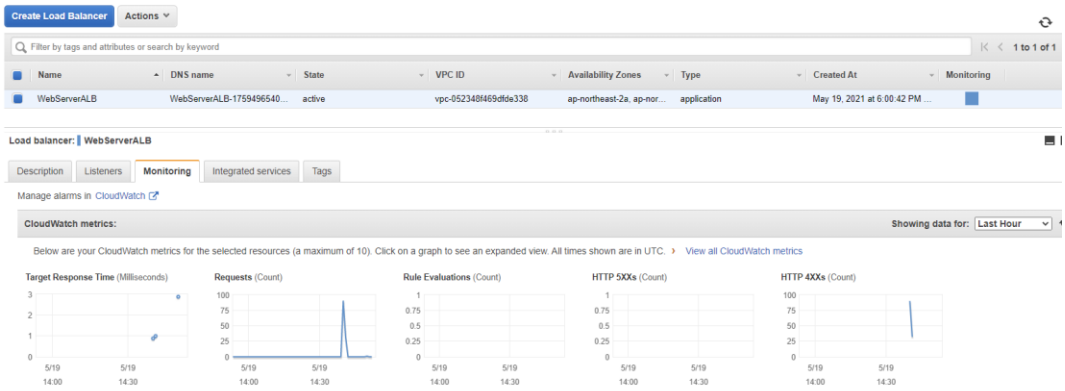
본격적인 복구 단계는 관리자가 AWS CLI를 활용해 복구를 시작하면서부터이다. 장애가 발생해 서비스가 중단된 Region이 감지되면 해당 장애 발생 Region이 아닌 다른 Region에 S3에 있는 각 Cloudformation Template를 업로드하면 재해복구가 시작된다. 모든 스택이 올라간 후나 급할 경우 Bastion이 생성 된 후, Bastion Host를 통하여 DB로 들어가 테이블을 작성한다.



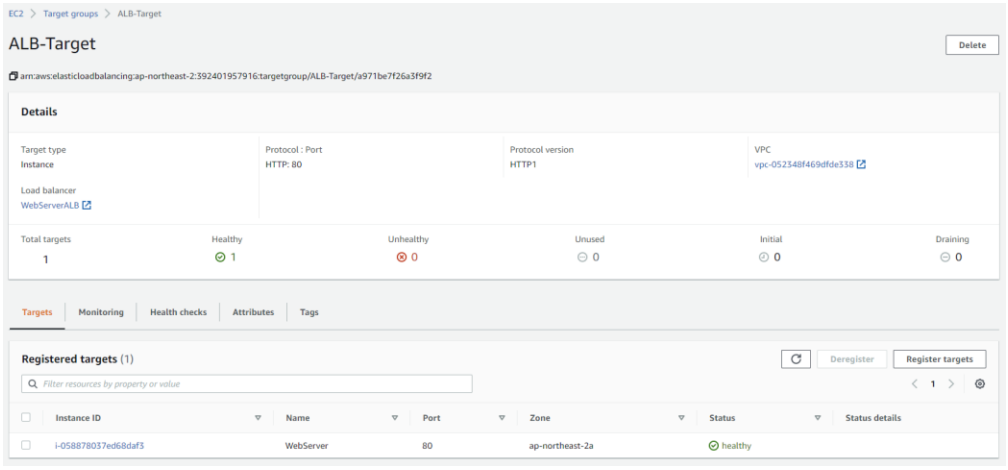
[그림 50] Bastion Host를 통한 DB테이블 세팅

6.4. 재해복구(DR) - 확인

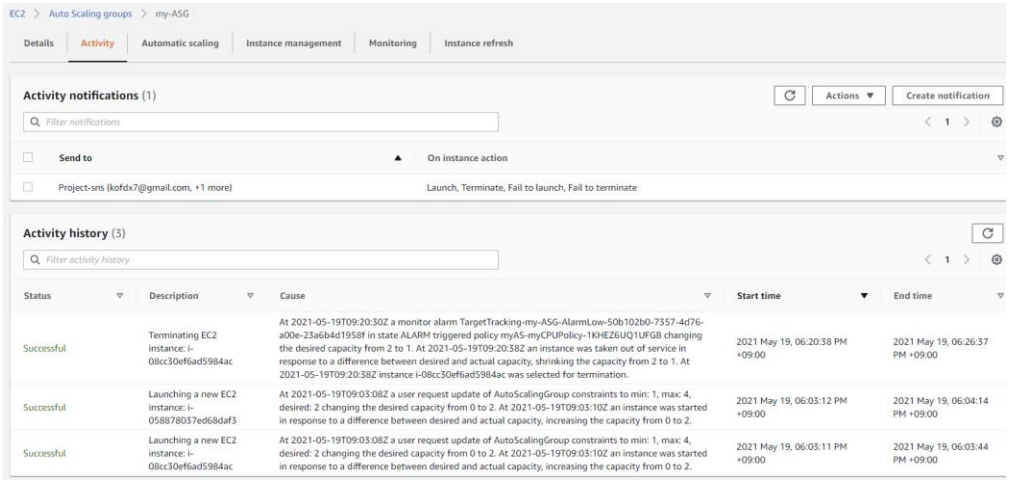
Golden AMI가 생성되면 NAT를 통해 외부통신이 가능한지 확인해야한다. apt-get install 등을 활용해 외부통신 여부를 확인한다. 또한 AS 스택이 생성되면 private subnet 1,2에 인스턴스가 알맞게 올라가는지 확인한다. 그 외에 Load Balancing 및 Auto scaling 작동과 회원가입 및 로그인을 통해 데이터베이스와의 연동 또한 확인해야 한다. 모든 확인이 끝나고 문제가 없을 시 재해복구가 마무리된다.



[그림 51] Load Balancing 확인 - monitoring



[그림 52] Load Balancing 확인 - target group



[그림 53] Auto Scaling 확인

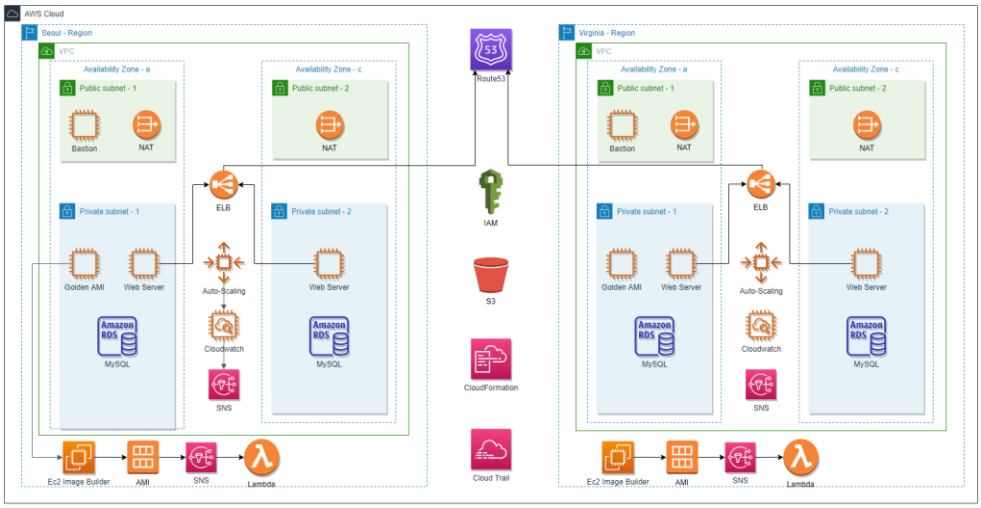
CloudFormation > Stacks

Stacks (7)

Filter by stack name | Active | View nested

Stack name	Status	Created time	Description
myAS	CREATE_COMPLETE	2021-05-20 10:44:42 UTC+0900	Create ALB, AS template
myGoldenAMI	CREATE_COMPLETE	2021-05-20 10:41:09 UTC+0900	GoldenAMI template
myNAT	CREATE_COMPLETE	2021-05-20 10:37:36 UTC+0900	NAT Instance template
myBastion	CREATE_COMPLETE	2021-05-20 10:31:46 UTC+0900	Bastion Host template
myRDS	CREATE_COMPLETE	2021-05-20 10:24:13 UTC+0900	RDS Template
myVPC	CREATE_COMPLETE	2021-05-20 10:23:09 UTC+0900	Create region1 VPC template

[그림 54] DR의 총 소요시간 (21분)



[그림 55] Virginia Region 복구 완료

7. 결과 및 개선 방안

7.1. 구현 결과

최종적으로 가용성과 확장성이 보장된 인프라 및 서비스 제공이 가능한 환경을 구축하였다. 웹 서버의 경우 실제 소비자가 회원가입하여 로그인/아웃하는 페이지 뿐만 아니라 희망 제품의 색상과 사이즈를 선택하여 주문이 가능하도록 구현하였다. 또한 직원의 권한 관리를 위해 IAM을 사용해 해당 조직의 역할에

따른 권한을 부여하였다. NAT와 Bastion Host를 제외한 서비스는 Private Subnet에 구축해 보안성을 높였으며 관계형 데이터베이스인 RDS를 사용해 관리의 용의성을 높였다. 트래픽을 분산시키기 위해 Load Balancers 서비스를 사용하였으며, Auto Scaling 서비스를 사용해 부하관리를 하였다. CloudWatch와 SNS 서비스를 사용해 관리자에게 전반적인 시스템 현황을 전달하도록 하였다.

DB의 Data 정리와 재해복구에 필요한 Cloudformation Template의 관리를 위해 S3 기능을 사용하였다. 재해복구는 IAM 권한을 가진 관리자가 AWS CLI를 사용하여 DR 목표 안으로 복구되도록 구현하였다.

7.2. 향후 개선 방안

프로젝트 진행 중 기본 기능을 테스트하기 위해 메인 페이지의 주요 기능만 구현하여 부하테스트 시 페이지나 기능별 트래픽 지정을 하지 못했다, 이러한 점을 보완하여 테스트를 진행할 수 있다면 더욱 기능적으로 정확한 지표를 나타내어 높은 신뢰성을 보장할 수 있을 것이라 생각한다.

8. 비용 산정

8.1. 예상비용

⊞	Amazon EC2 Service (Asia Pacific (Osaka))	$ 362.55
⊞	Amazon EC2 Service (Asia Pacific (Seoul))	$ 502.10
⊞	Amazon S3 Service (Asia Pacific (Osaka))	$ 2.50
⊞	Amazon S3 Service (Asia Pacific (Seoul))	$ 0.03
⊞	Amazon Route 53 Service	$ 0.50
⊞	Amazon RDS Service (Asia Pacific (Osaka))	$ 3186.16
⊞	Amazon RDS Service (Asia Pacific (Seoul))	$ 2914.52
⊞	Amazon Elastic Load Balancing (Asia Pacific (Osaka))	$ 19.77
⊞	Amazon Elastic Load Balancing (Asia Pacific (Seoul))	$ 34.77
	Amazon CloudWatch Service (Asia Pacific (Osaka))	$ 0.00
	Amazon CloudWatch Service (Asia Pacific (Seoul))	$ 0.00
⊞	Amazon SNS Service (Asia Pacific (Osaka))	$ 0.00
⊞	Amazon SNS Service (Asia Pacific (Seoul))	$ 0.00
⊞	Amazon VPC Service (Asia Pacific (Osaka))	$ 35.14
⊞	Amazon VPC Service (Asia Pacific (Seoul))	$ 36.60
⊞	AWS Data Transfer In	$ 0.00
⊞	AWS Data Transfer Out	$ 0.00
⊞	AWS Support (Business)	$ 707.24
Free Tier Discount:		**$** -22.30
Total Monthly Payment:		**$** 7779.58

[그림 56] 예상 비용

9. 부록

9.1. CloudFormation Template Code - YAML

9.1.1. Seoul Region

[부록 1] VPC.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create region1 VPC template


Parameters:


  LabVpcCidr:
```

```
    Type: String
    Default: 10.5.0.0/16

  PublicSubnet1Cidr:
    Type: String
    Default: 10.5.10.0/24

  PublicSubnet2Cidr:
    Type: String
    Default: 10.5.30.0/24

  PrivateSubnet1Cidr:
    Type: String
    Default: 10.5.20.0/24

  PrivateSubnet2Cidr:
    Type: String
    Default: 10.5.40.0/24


Resources:

############
# VPC with Internet Gateway
############

  LabVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref LabVpcCidr
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
```

```
        - Key: Name
          Value: Lab VPC

  IGW:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: Lab IGW

  VPCtoIGWConnection:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref IGW
      VpcId: !Ref LabVPC

############
# Public Route Table
############

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref LabVPC
      Tags:
        - Key: Name
          Value: Public Route Table

  PublicRoute:
    Type: AWS::EC2::Route
    Properties:
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref IGW
```

```
      RouteTableId: !Ref PublicRouteTable

############
# Public Subnet
############

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref LabVPC
      MapPublicIpOnLaunch: true
      CidrBlock: !Ref PublicSubnet1Cidr
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Public Subnet1

  PublicSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref LabVPC
      MapPublicIpOnLaunch: true
      CidrBlock: !Ref PublicSubnet2Cidr
      AvailabilityZone: !Select
        - 2
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Public Subnet2
```

```
  PublicSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnet1
      RouteTableId: !Ref PublicRouteTable

  PublicSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PublicSubnet2
      RouteTableId: !Ref PublicRouteTable


############
# Private Route Table
############

  PrivateRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref LabVPC
      Tags:
        - Key: Name
          Value: Private Route Table

############
# Private Subnet
############

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
```

```
      VpcId: !Ref LabVPC
      CidrBlock: !Ref PrivateSubnet1Cidr
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Private1 Subnet

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref LabVPC
      CidrBlock: !Ref PrivateSubnet2Cidr
      AvailabilityZone: !Select
        - 2
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Private2 Subnet

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet1
      RouteTableId: !Ref PrivateRouteTable

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet2
```

```
      RouteTableId: !Ref PrivateRouteTable


Outputs:
Outputs:
  PrivateSubnet1:
    Value: !Ref PrivateSubnet1
    Description: ID of PrivateSubnet1
    Export:
      Name: !Sub ${AWS::StackName}-PrivateSubnet1

  PrivateSubnet2:
    Value: !Ref PrivateSubnet2
    Description: ID of PrivateSubnet2
    Export:
      Name: !Sub ${AWS::StackName}-PrivateSubnet2

  LabVPC:
    Value: !Ref LabVPC
    Description: ID of LabVPC
    Export:
      Name: !Sub ${AWS::StackName}-LabVPC

  PublicSubnet1:
    Value: !Ref PublicSubnet1
    Description: ID of PublicSubnet1
    Export:
      Name: !Sub ${AWS::StackName}-PublicSubnet1

  PublicSubnet2:
    Value: !Ref PublicSubnet2
    Description: ID of PublicSubnet2
    Export:
```

```
    Name: !Sub ${AWS::StackName}-PublicSubnet2

PublicSubnet1Cidr:
  Value: !Ref PublicSubnet1Cidr
  Description: ID of PublicSubnet1
  Export:
    Name: !Sub ${AWS::StackName}-PublicSubnet1Cidr

PublicSubnet2Cidr:
  Value: !Ref PublicSubnet2Cidr
  Description: ID of PublicSubnet2
  Export:
    Name: !Sub ${AWS::StackName}-PublicSubnet2Cidr

PrivateSubnet1Cidr:
  Value: !Ref PrivateSubnet1Cidr
  Description: ID of PublicSubnet1
  Export:
    Name: !Sub ${AWS::StackName}-PrivateSubnet1Cidr

PrivateSubnet2Cidr:
  Value: !Ref PublicSubnet2Cidr
  Description: ID of PrivateSubnet2Cidr
  Export:
    Name: !Sub ${AWS::StackName}-PrivateSubnet2Cidr


PublicRouteTable:
  Value: !Ref PublicRouteTable
  Description: ID of PublicRoute
  Export:
    Name: !Sub ${AWS::StackName}-PublicRouteTable
```

```
  PrivateRouteTable:
    Value: !Ref PrivateRouteTable
    Description: ID of PrivateRouteTable
    Export:
      Name: !Sub ${AWS::StackName}-PrivateRouteTable

  VPCDefaultSG:
    Value: !GetAtt LabVPC.DefaultSecurityGroup
    Description: ID of VPC default sg
    Export:
      Name: !Sub ${AWS::StackName}-VPCDefaultSG
```

[부록 2] RDS.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: RDS Template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values VPC
    Type: String
  DBUsername:
    Description: DB Username
    Default: admin
    NoEcho: true
    Type: String
    MinLength: '1'
    MaxLength: '16'
  DBPassword:
    Description: DB Pwd
```

```
    NoEcho: true
    Type: String
    MinLength: '8'
    MaxLength: '41'

  DBInstanceIdentifier:
    Description: Instance Identifier name
    Type: String

Resources:

############
# RDS
############


  RDSInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      DBName: DBName
      Engine: MySQL
      EngineVersion: 8.0.20
      MasterUsername: !Ref DBUsername
      MasterUserPassword: !Ref DBPassword
      DBInstanceClass: db.t2.micro
      VPCSecurityGroups:
        -  Fn::GetAtt: [ RDSsg, GroupId ]
      AllocatedStorage: 20
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      StorageType: gp2
```

```
      DBInstanceIdentifier: !Ref DBInstanceIdentifier
      PubliclyAccessible: false
      DBSubnetGroupName: !Ref RDSSubnetGroup

###########
# RDS Subnet Group
###########

  RDSSubnetGroup:
    Type: "AWS::RDS::DBSubnetGroup"
    Properties:
      DBSubnetGroupName: RDS Private Subnet Group
      DBSubnetGroupDescription: RDS Subnet Group
      SubnetIds:
        - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1'
        - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet2'
      Tags:
        -
          Key: Name
          Value: RDSSubnetGroup

###########
# RDS sg
###########

  RDSsg:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: RDSsg
      GroupDescription: Open Port 3306
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 3306
```

```
        ToPort: 3306
        CidrIp:
          Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1Cidr'
      - IpProtocol: tcp
        FromPort: 3306
        ToPort: 3306
        CidrIp:
          Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1Cidr'
      - IpProtocol: tcp
        FromPort: 3306
        ToPort: 3306
        CidrIp:
          Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet2Cidr'

    Tags:
      -
        Key: Name
        Value: RDSsg
    VpcId:
      Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'


Outputs:

  RDSEndPointAddress:
    Value: !GetAtt RDSInstance.Endpoint.Address
    Description: RDS EndPoint Address
    Export:
      Name: !Sub ${AWS::StackName}-RDSEndPointAddress
```

[부록 3] Bastion.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: Bastion Host template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  BastionAMIID:
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
    Default:
/aws/service/canonical/ubuntu/server/18.04/stable/20210224/amd64/hvm/ebs-
gp2/ami-id




Resources:

############
# Bastion
###########

  BastionInstance:
    Type: AWS::EC2::Instance
    Properties:
```

```
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref BastionAMIID
      SubnetId:
        Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
      SecurityGroupIds:
        - !Ref BastionSecurityGroup
      IamInstanceProfile: !Ref InstanceProfile
      Tags:
        - Key: Name
          Value: Bastion Host
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          apt update
          apt install -y awscli
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
          apt install unzip
          unzip awscliv2.zip
          sudo ./aws/install

          apt install -y mysql-client

############
# Bastion SG
############

  BastionSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: Bastion
      GroupDescription: Open Port 22 for ssh
```

```
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: Bastion


############
# IAM Role for Instance
############

  KrBastionRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: KrBastionEc2FullAccess
          PolicyDocument:
            Version: "2012-10-17"
```

```
            Statement:
              - Effect: Allow
                Action: 'ec2:*'
                Resource: '*'
              - Effect: Allow
                Action: 'cloudformation:ListExports'
                Resource: '*'
              - Effect: Allow
                Action: 'cloudformation:DescribeStacks'
                Resource: '*'
              - Effect: Allow
                Action: 'rds:*'
                Resource: '*'
              - Effect: Allow
                Action: 's3:*'
                Resource: '*'

############
# Instance Profile
############

  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref KrBastionRole]
      InstanceProfileName: KrBastionRole



Outputs:
  BastionPublicDns:
    Value: !GetAtt BastionInstance.PublicDnsName
```

```
    Description: Bastion Dns Name
    Export:
      Name: !Sub ${AWS::StackName}-BastionPublicDns

  BastionId:
    Value: !Ref BastionInstance
    Description: Bastion Instance Id
    Export:
      Name: !Sub ${AWS::StackName}-BastionId

  BastionIp:
    Value: !GetAtt BastionInstance.PrivateIp
    Description: BastionInstance Instance ip
    Export:
      Name: !Sub ${AWS::StackName}-BastionInstanceIp
```

[부록 4] NAT.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: NAT Instance template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values VPCs
    Type: String

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  UbuntuAMI:
```

```
    Type: String
    Description: NAT Instance AMI ID
    Default: ami-0ba5cd124d7a79612



Resources:

############
# NAT Instance
###########

  NatInstance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref UbuntuAMI
      SubnetId:
        Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
      SecurityGroupIds:
        - !Ref NATSecurityGroup
      IamInstanceProfile: !Ref InstanceProfile
      Tags:
        - Key: Name
          Value: NAT
      SourceDestCheck: False
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash

          sudo apt update -y
```

```
          sudo apt install -y awscli
          sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
          sudo apt install unzip
          sudo unzip awscliv2.zip
          sudo ./aws/install

          sudo apt-get update
          echo iptables-persistent iptables-persistent/autosave_v4 boolean true | sudo
debconf-set-selections
          echo iptables-persistent iptables-persistent/autosave_v6 boolean true | sudo
debconf-set-selections
          sudo apt-get -y install iptables-persistent
          echo 1 > /proc/sys/net/ipv4/ip_forward

          sudo sed -i '28s/#//g' /etc/sysctl.conf

          iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
          iptables -t nat -A POSTROUTING -o eth0 -s 0.0.0.0/0 -j MASQUERADE

          iptables -t nat -L
          netfilter-persistent save
          cat /etc/iptables/rules.v4


############
# NAT SG
############

  NATSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: Nat
```

```
      GroupDescription: Open Port 22 for ssh
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 0
          ToPort: 65535
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: NAT


############
# IAM Role for Instance
############

  krNatRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: krNatFullAccess
          PolicyDocument:
```

```
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'ec2:*'
                Resource: '*'


###########
# Instance Profile
###########

  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref krNatRole]
      InstanceProfileName: krNatRole



#############
#PrivateRoute
#############


  PrivateRoute:
    Type: AWS::EC2::Route
    Properties:
      DestinationCidrBlock: 0.0.0.0/0
      InstanceId: !Ref NatInstance
      RouteTableId:
        Fn::ImportValue: !Sub '${ExportStackName}-PrivateRouteTable'
```

```
Outputs:
  NATPublicDns:
    Value: !GetAtt NatInstance.PublicDnsName
    Description: NAT Dns Name
    Export:
      Name: !Sub ${AWS::StackName}-NATDns

  NATId:
    Value: !Ref NatInstance
    Description: NAT Instance Id
    Export:
      Name: !Sub ${AWS::StackName}-NATId

  NATIp:
    Value: !GetAtt NatInstance.PrivateIp
    Description: NAT Instance ip
    Export:
      Name: !Sub ${AWS::StackName}-NATIp
```

[부록 5] Golden_AMI.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: GoldenAMI template


Parameters:
  ExportStackName:
```

```
    Description: The name of the stack that exports the values
    Type: String

  KeyName:
    Type: String
    Description: Select KeyPair
    Default: krAdmin

  GoldenAMIID:
    Type: String
    Description: This AMI id is WEBv3
    Default: ami-0c3875b921d298b30


Resources:

############
# GoldenAMI
##########

  GoldenAMI:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref GoldenAMIID
      SubnetId:
        Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1'
      SecurityGroupIds:
        - !Ref GoldenAMISecurityGroup
      IamInstanceProfile: !Ref InstanceProfile
      Tags:
        - Key: Name
```

```
          Value: Golden AMI
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          ####install aws cli
          apt update
          apt install -y awscli
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
          apt install unzip
          unzip awscliv2.zip
          sudo ./aws/install
          ####install aws docker and sam cli
          sudo apt install -y docker.io
          sudo systemctl start docker
          aws s3 cp s3://golden-ami-suyeonshoes/aws-sam-cli-linux-x86_64.zip ./
          sudo unzip aws-sam-cli-linux-x86_64.zip -d sam-installation
          sudo ./sam-installation/install

          ####deploy AS stack
          aws s3 cp s3://golden-ami-suyeonshoes/AS_Region1.zip ./
          sudo unzip AS_Region1.zip
          sam build --use-container
          sam package --output-template-file packaged.yaml --s3-bucket golden-ami-
suyeonshoes --region ap-northeast-2
          sam deploy --template-file packaged.yaml --stack-name myAS --capabilities
CAPABILITY_IAM --region ap-northeast-2


############
# Golden AMI SG
############
```

```
  GoldenAMISecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: GoldenAMI
      GroupDescription: Open Port 22 for ssh
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1Cidr'
      Tags:
        - Key: Name
          Value: GoldenAMISecurityGroup


############
# IAM Role for Instance
############

  krGoldenAMIRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
```

```
        - sts:AssumeRole
Path: /
Policies:
  - PolicyName: krGoldenAMIEc2FullAccess
    PolicyDocument:
      Version: "2012-10-17"
      Statement:
        - Effect: Allow
          Action: 'ec2:*'
          Resource: '*'
        - Effect: Allow
          Action: 'cloudformation:*'
          Resource: '*'
        - Effect: Allow
          Action: 'iam:*'
          Resource: '*'
        - Effect: Allow
          Action: 's3:*'
          Resource: '*'
        - Effect: Allow
          Action: 'imagebuilder:*'
          Resource: '*'
        - Effect: Allow
          Action: 'ssm:*'
          Resource: '*'
        - Effect: Allow
          Action: 'lambda:*'
          Resource: '*'
        - Effect: Allow
          Action: 'autoscaling:*'
          Resource: '*'
        - Effect: Allow
          Action: 'sts:*'
```

```
                Resource: '*'
              - Effect: Allow
                Action: 'sns:*'
                Resource: '*'
              - Effect: Allow
                Action: 'elasticloadbalancing:*'
                Resource: '*'
              - Effect: Allow
                Action: 'rds:*'
                Resource: '*'


############
# Instance Profile
############

  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref krGoldenAMIRole]
      InstanceProfileName: krGoldenAMIRole



Outputs:
  GoldenAMIaddr:
    Value: !GetAtt GoldenAMI.PrivateIp
    Description: GoldenAMIaddr
    Export:
      Name: !Sub ${AWS::StackName}-GoldenAMIaddr

  GoldenAMIId:
```

```
    Value: !Ref GoldenAMI
    Description: GoldenAMI Instance Id
    Export:
      Name: !Sub ${AWS::StackName}-GoldenAMIId
```

[부록 6] template.yaml(Auto Scaling, EC2 Image Builder, SNS, Lambda)

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'

Metadata:
  License:
    Description: >
      Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
      SPDX-License-Identifier: MIT-0
      AutoScalingGroupInstanceType
      Permission is hereby granted, free of charge, to any person obtaining a copy of this
      software and associated documentation files (the "Software"), to deal in the Software
      without restriction, including without limitation the rights to use, copy, modify,
      merge, publish, distribute, sublicense, and/or sell copies of the Software, and to
      permit persons to whom the Software is furnished to do so.

      THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED,
      INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A
      PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
      HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
```

```
WHETHER IN AN ACTION
      OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE
      SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String
    Default: "myVPC"
  EnvironmentName:
    Type: String
    Default: "Web Server"
  AmiId:
    Type: String
    Default: "ami-074636aec5e81a80a"
  BuildInstanceType:
    Type: String
    Default: "t2.micro"
    Description: "Image Builder instance type"
  AutoScalingGroupInstanceType:
    Type: String
    Default: "t2.micro"
    Description: Instance type for sample Auto Scaling groupi
  KeyName:
    Type: String
    Default: "krAdmin"
  Email:
    Type: String
    Default: "kofdx7@gmail.com"

Resources:
```

```
############
# InstanceRefreshHandler
############
  InstanceRefreshHandler:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: lambda_function.lambda_handler
      Runtime: python3.7
      MemorySize: 128
      Timeout: 30
      Role: !GetAtt InstanceRefreshHandlerLambdaRole.Arn
      CodeUri: InstanceRefreshHandler/
      Environment:
        Variables:
          AutoScalingGroupName: !Ref AutoScalingGroup


############
# InstanceRefreshHandlerLambdaRole
############
  InstanceRefreshHandlerLambdaRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
```

```
      Path: "/service-role/"
      Policies:
      - PolicyName: krlambdaExecution-InstanceRefreshHandler
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            Resource: '*'
          - Effect: Allow
            Action:
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
            - autoscaling:StartInstanceRefresh
            - autoscaling:Describe*
            - ec2:CreateLaunchTemplateVersion
            - ec2:DescribeLaunchTemplates
            Resource: '*'

###########
# ImageBuilderSNSTopic
###########
  ImageBuilderSNSTopic:
    Type: "AWS::SNS::Topic"
    Properties:
      Subscription:
        - Endpoint: !GetAtt InstanceRefreshHandler.Arn
          Protocol: lambda
###########
```

```
# SNSLambdaPermission
############

  SNSLambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
        FunctionName: !GetAtt InstanceRefreshHandler.Arn
        Action: lambda:InvokeFunction
        Principal: sns.amazonaws.com
        SourceArn: !Ref ImageBuilderSNSTopic


############
# EC2ImageBuilderRecipe
############
  EC2ImageBuilderRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
      Name: EC2ImageBuilderRecipe
      Description: This recipe updates the system and installs Docker CE
      ParentImage: !Ref AmiId
      Components:
        - ComponentArn: !Sub
"arn:aws:imagebuilder:${AWS::Region}:aws:component/update-linux/1.0.0/1"
        - ComponentArn: !Sub
"arn:aws:imagebuilder:${AWS::Region}:aws:component/reboot-linux/1.0.1/1"
      Version: "1.0.0"

############
# EC2ImageBuilderPipeline
############
  EC2ImageBuilderPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
```

```
    Properties:
      Name: EC2ImageBuilderPipeline
      ImageRecipeArn: !Ref EC2ImageBuilderRecipe
      InfrastructureConfigurationArn: !Ref EC2ImageBuilderInfrastructureConfiguration
      DistributionConfigurationArn: !Ref DistributionConfigurationArn


###########
# EC2ImageBuilder DistributionConfiguration
###########
  DistributionConfiguration:
    Type: AWS::ImageBuilder::DistributionConfiguration
    Properties:
      Name: 'distribution-configuration-name'
      Description: 'description'
      Distributions:
        - Region: 'ap-northeast-2'
          AmiDistributionConfiguration:
            Name: 'distribution korea {{ imagebuilder:buildDate }}'
        - Region: 'us-east-1'
          AmiDistributionConfiguration:
            Name: 'distribution backup {{ imagebuilder:buildDate }}'
      Tags:
        CustomerDistributionConfigTagKey1: 'CustomerDistributionConfigTagValue1'
        CustomerDistributionConfigTagKey2: 'CustomerDistributionConfigTagValue2'




###########
# EC2ImageBuilderInfrastructureConfiguration
###########
  EC2ImageBuilderInfrastructureConfiguration:
```

```
    Type: AWS::ImageBuilder::InfrastructureConfiguration
    Properties:
      Name: InstanceConfigurationForEC2ImageBuilder
      InstanceTypes:
        - !Ref BuildInstanceType
      InstanceProfileName: !Ref EC2ImageBuilderIAMInstanceProfile
      SNSTopicArn: !Ref ImageBuilderSNSTopic
      SubnetId:
         Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
      SecurityGroupIds:
        - !Ref ASTemplateGroup
      TerminateInstanceOnFailure: true

############
# EC2ImageBuilderIAMRole
############
  EC2ImageBuilderIAMRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/EC2InstanceProfileForImageBuilder
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
      Policies:
        - PolicyName: krEC2ImageBuilderIAM
```

```
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'cloudformation:ListExports'
                Resource: '*'
              - Effect: Allow
                Action: 'cloudformation:DescribeStacks'
                Resource: '*'


###########
# EC2ImageBuilderIAMInstanceProfile
###########
  EC2ImageBuilderIAMInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
        - !Ref EC2ImageBuilderIAMRole


###########
# AutoScaling Group
###########
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MinSize: "1"
      MaxSize: "4"
      DesiredCapacity: "2"
      HealthCheckGracePeriod: 180
      HealthCheckType: ELB
      TargetGroupARNs:
```

```
      - !Ref LoadBalancerTargetGroup
    VPCZoneIdentifier:
      - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1'
      - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet2'
    MetricsCollection:
      - Granularity: "1Minute"
    CapacityRebalance: true
    NotificationConfigurations:
    - TopicARN: !Ref SnsTopic
      NotificationTypes:
      - autoscaling:EC2_INSTANCE_LAUNCH
      - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
      - autoscaling:EC2_INSTANCE_TERMINATE
      - autoscaling:EC2_INSTANCE_TERMINATE_ERROR
      - autoscaling:TEST_NOTIFICATION
    LaunchTemplate:
      LaunchTemplateId: !Ref LaunchTemplate
      Version: !GetAtt LaunchTemplate.LatestVersionNumber
    Tags:
      - Key: Name
        Value: !Ref EnvironmentName
        PropagateAtLaunch: true

############
# AS-CPU-Policy
############
  myCPUPolicy:
    Type: AWS::AutoScaling::ScalingPolicy
    Properties:
      AutoScalingGroupName: !Ref AutoScalingGroup
      PolicyType: TargetTrackingScaling
      TargetTrackingConfiguration:
        PredefinedMetricSpecification:
```

```
          PredefinedMetricType: ASGAverageCPUUtilization
        TargetValue: 60.00


###########
# SNS-Topic
###########
  SnsTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: topic-sns
      FifoTopic: False
      TopicName: Project-sns

###########
# SNS-Subscription
###########
  SnsSub:
    Type: AWS::SNS::Subscription
    Properties:
      Endpoint: !Ref Email
      Protocol: Email
      TopicArn: !Ref SnsTopic


  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId: !Ref AmiId
        KeyName: !Ref KeyName
        IamInstanceProfile:
          Arn: !GetAtt EC2ImageBuilderIAMInstanceProfile.Arn
```

```
        InstanceType: !Ref AutoScalingGroupInstanceType
        SecurityGroupIds:
          - !Ref ASTemplateGroup
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            sudo apt update -y
            sudo apt install -y awscli
            sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
            sudo apt install unzip
            sudo unzip awscliv2.zip
            sudo ./aws/install

            dbaddr='$mysql_hostname='"'"`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`RDSEndPointAddress₩`].OutputValue' --output=text
--region=ap-northeast-2`"';"
            sudo sed -i "4s/.*/$dbaddr/g" /var/www/html/basic/login/dbconn.php

############
# Application Load Balancer
############
  LoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Type: application
      Name: WebServerALB
      Scheme: internet-facing
      SubnetMappings:
        - SubnetId:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
        - SubnetId:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet2'
```

```
      IpAddressType: ipv4
      SecurityGroups:
        - !Ref ASTemplateGroup

###########
# Application Load Balancer TargetGroup
###########
  LoadBalancerTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Name: ALB-Target
      TargetType: instance
      HealthyThresholdCount: 2
      HealthCheckIntervalSeconds: 30
      TargetGroupAttributes:
        - Key: stickiness.enabled
          Value: true
        - Key: stickiness.type
          Value: lb_cookie
        - Key: stickiness.lb_cookie.duration_seconds
          Value: 86400
      Protocol: HTTP
      Port: 80
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'


###########
# Application Load Balancer Listener
###########
  LoadBalancerListener:
   Type: AWS::ElasticLoadBalancingV2::Listener
   Properties:
```

```
      DefaultActions:
        - Type: "forward"
          ForwardConfig:
            TargetGroups:
             - TargetGroupArn: !Ref LoadBalancerTargetGroup
      LoadBalancerArn: !Ref LoadBalancer
      Port: 80
      Protocol: "HTTP"


############
# Auto Scaling Temp Security Group
############

  ASTemplateGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: ASTemplate
      GroupDescription: ASG Template Security Group
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: ASTemplate
```

```
Outputs:
  EC2ImageBuilderPipeline:
    Description: Sample EC2 Image Builder Pipeline
    Value: !Ref EC2ImageBuilderPipeline
  SNSTopic:
    Description: Amazon SNS topic subscribed to the EC2 Image Builder pipeline to
trigger Lambda
    Value: !Ref ImageBuilderSNSTopic
  LambdaFunction:
    Description: AWS Lambda function handling EC2 Image Builder Notifications and
triggering Auto Scaling Instance Refresh
    Value: !Ref InstanceRefreshHandler
  AutoScalingGroup:
    Description: Sample Auto Scaling group
    Value: !Ref AutoScalingGroup
  LaunchTemplate:
    Description: Sample Launch Template for Auto Scaling group
    Value: !Ref LaunchTemplate
  ALBDns:
    Value: !GetAtt LoadBalancer.DNSName
    Description: Load Balancer Dns
    Export:
      Name: !Sub ${AWS::StackName}-LoadBalancerDns
  ALBHz:
    Value: !GetAtt LoadBalancer.CanonicalHostedZoneID
    Description: Load Balancer HZ
    Export:
      Name: !Sub ${AWS::StackName}-LoadBalancerHZ
```

[부록 7] Route53.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: Route53 template

Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String

  KrALBDns:
    Description: The name of the stack that exports the values
    Type: String

  KrALBHz:
    Description: The name of the stack that exports the values
    Type: String
  HzId:
    Description: The name of the stack that exports the values
    Type: String
    Default: Z02045551T74ATR5QH0EX


Resources:

  Server:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Register DNS
      HostedZoneId:
        !Ref HzId
      RecordSets:
      - Name: www.suyeonshoes.click
        Type: A
```

```
      AliasTarget:
        HostedZoneId:
          !Ref KrALBHz
        DNSName:
          !Ref KrALBDns
    - Name: kr.suyeonshoes.click
      Type: A
      AliasTarget:
        HostedZoneId:
          !Ref KrALBHz
        DNSName:
          !Ref KrALBDns
```

[부록 8] CloudTrail.yaml

```
  AWSTemplateFormatVersion: 2010-09-09
Description: Test template

Parameters:
  OperatorEmail:
    Description: "Email address to notify when new logs are published."
    Type: String
    Default: mgk9530@gmail.com

Resources:

############
# CloudTrail
############
  cloudTrail:
    Type: AWS::CloudTrail::Trail
```

```
    Properties:
      CloudWatchLogsLogGroupArn:
        Fn::GetAtt:
          - "logGroup"
          - "Arn"
      CloudWatchLogsRoleArn:
        Fn::GetAtt:
          - "IamRoleForCwLogsCloudTrail"
          - "Arn"
      EnableLogFileValidation: True
      EventSelectors:
        - IncludeManagementEvents: True
      IncludeGlobalServiceEvents: True
      IsLogging: True
      IsMultiRegionTrail: True
      S3BucketName: !Ref s3fortrail
      SnsTopicName:
        Fn::GetAtt:
          - Topic
          - TopicName
      TrailName: test-Trail

###############
# S3 bucket for CWT
################
  s3fortrail:
    Type: AWS::S3::Bucket
    Properties: {}

###############
# S3 policy for CWT
###############
  S3BucketPolicy:
```

```
Type: AWS::S3::BucketPolicy
Properties:
  Bucket:
    !Ref s3fortrail
  PolicyDocument:
    Version: "2012-10-17"
    Statement:
      - Sid: AWSCloudTrailBucketPermissionsCheck
        Effect: Allow
        Principal:
          Service:
            - cloudtrail.amazonaws.com
        Action: s3:GetBucketAcl
        Resource:
          Fn::GetAtt:
            - s3fortrail
            - Arn
      - Sid: AWSConfigBucketDelivery
        Effect: Allow
        Principal:
          Service:
            - "cloudtrail.amazonaws.com"
        Action: "s3:PutObject"
        Resource:
          Fn::Join:
            - ""
            -
              - Fn::GetAtt:
                  - "s3fortrail"
                  - "Arn"
              - "/AWSLogs/*"
        Condition:
          StringEquals:
```

```
                s3:x-amz-acl: "bucket-owner-full-control"

############
#Topic for CWT
############
  Topic:
    Type: AWS::SNS::Topic
    Properties:
      Subscription:
        -
          Endpoint:
            Ref: OperatorEmail
          Protocol: email

##################
# Topic-policy for CLT
##################
  TopicPolicy:
    Type: AWS::SNS::TopicPolicy
    Properties:
      Topics:
        - Ref: Topic
      PolicyDocument:
        Version: "2008-10-17"
        Statement:
          -
            Sid: "AWSCloudTrailSNSPolicy"
            Effect: "Allow"
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Resource: "*"
            Action: "SNS:Publish"
```

```
############
# Loggroup
############
  logGroup:
    Type: AWS::Logs::LogGroup
    Properties:
      LogGroupName: logGroupForTrail
      RetentionInDays: 1

###############
# IAM Role for CWT
###############
  IamRoleForCwLogsCloudTrail:
    Type: "AWS::IAM::Role"
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Sid: ""
            Effect: "Allow"
            Principal:
              Service: "cloudtrail.amazonaws.com"
            Action: "sts:AssumeRole"
      Policies:
        - PolicyName: "allow-access-to-cw-logs"
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: "Allow"
                Action:
                  - "logs:CreateLogStream"
                  - "logs:PutLogEvents"
```

```
            Resource: "*"
```

9.1.2. Osaka Region

[부록 9] VPC.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: Create VPC template


Parameters:

  LabVpcCidr:
    Type: String
    Default: 192.168.0.0/16

  PublicSubnet1Cidr:
    Type: String
    Default: 192.168.10.0/24

  PublicSubnet2Cidr:
    Type: String
    Default: 192.168.30.0/24

  PrivateSubnet1Cidr:
    Type: String
    Default: 192.168.20.0/24
```

```
  PrivateSubnet2Cidr:
    Type: String
    Default: 192.168.40.0/24


Resources:

############
# VPC with Internet Gateway
############

  LabVPC:
    Type: AWS::EC2::VPC
    Properties:
      CidrBlock: !Ref LabVpcCidr
      EnableDnsSupport: true
      EnableDnsHostnames: true
      Tags:
        - Key: Name
          Value: Lab VPC

  IGW:
    Type: AWS::EC2::InternetGateway
    Properties:
      Tags:
        - Key: Name
          Value: Lab IGW

  VPCtoIGWConnection:
    Type: AWS::EC2::VPCGatewayAttachment
    Properties:
      InternetGatewayId: !Ref IGW
```

```
      VpcId: !Ref LabVPC

############
# Public Route Table
############

  PublicRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref LabVPC
      Tags:
        - Key: Name
          Value: Public Route Table

  PublicRoute:
    Type: AWS::EC2::Route
    Properties:
      DestinationCidrBlock: 0.0.0.0/0
      GatewayId: !Ref IGW
      RouteTableId: !Ref PublicRouteTable

############
# Public Subnet
############

  PublicSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref LabVPC
      MapPublicIpOnLaunch: true
      CidrBlock: !Ref PublicSubnet1Cidr
      AvailabilityZone: !Select
        - 0
```

```
      - !GetAZs
        Ref: AWS::Region
    Tags:
      - Key: Name
        Value: Public Subnet1

PublicSubnet2:
  Type: AWS::EC2::Subnet
  Properties:
    VpcId: !Ref LabVPC
    MapPublicIpOnLaunch: true
    CidrBlock: !Ref PublicSubnet2Cidr
    AvailabilityZone: !Select
      - 2
      - !GetAZs
        Ref: AWS::Region
    Tags:
      - Key: Name
        Value: Public Subnet2

PublicSubnet1RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet1
    RouteTableId: !Ref PublicRouteTable

PublicSubnet2RouteTableAssociation:
  Type: AWS::EC2::SubnetRouteTableAssociation
  Properties:
    SubnetId: !Ref PublicSubnet2
    RouteTableId: !Ref PublicRouteTable
```

```
############
# Private Route Table
############

  PrivateRouteTable:
    Type: AWS::EC2::RouteTable
    Properties:
      VpcId: !Ref LabVPC
      Tags:
        - Key: Name
          Value: Private Route Table

############
# Private Subnet
############

  PrivateSubnet1:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref LabVPC
      CidrBlock: !Ref PrivateSubnet1Cidr
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Private1 Subnet

  PrivateSubnet2:
    Type: AWS::EC2::Subnet
    Properties:
      VpcId: !Ref LabVPC
```

```
      CidrBlock: !Ref PrivateSubnet2Cidr
      AvailabilityZone: !Select
        - 2
        - !GetAZs
          Ref: AWS::Region
      Tags:
        - Key: Name
          Value: Private2 Subnet

  PrivateSubnet1RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet1
      RouteTableId: !Ref PrivateRouteTable

  PrivateSubnet2RouteTableAssociation:
    Type: AWS::EC2::SubnetRouteTableAssociation
    Properties:
      SubnetId: !Ref PrivateSubnet2
      RouteTableId: !Ref PrivateRouteTable



Outputs:
Outputs:
  PrivateSubnet1:
    Value: !Ref PrivateSubnet1
    Description: ID of PrivateSubnet1
    Export:
      Name: !Sub ${AWS::StackName}-PrivateSubnet1

  PrivateSubnet2:
    Value: !Ref PrivateSubnet2
    Description: ID of PrivateSubnet2
```

```
  Export:
    Name: !Sub ${AWS::StackName}-PrivateSubnet2

LabVPC:
  Value: !Ref LabVPC
  Description: ID of LabVPC
  Export:
    Name: !Sub ${AWS::StackName}-LabVPC

PublicSubnet1:
  Value: !Ref PublicSubnet1
  Description: ID of PublicSubnet1
  Export:
    Name: !Sub ${AWS::StackName}-PublicSubnet1

PublicSubnet2:
  Value: !Ref PublicSubnet2
  Description: ID of PublicSubnet2
  Export:
    Name: !Sub ${AWS::StackName}-PublicSubnet2

PublicSubnet1Cidr:
  Value: !Ref PublicSubnet1Cidr
  Description: ID of PublicSubnet1
  Export:
    Name: !Sub ${AWS::StackName}-PublicSubnet1Cidr

PublicSubnet2Cidr:
  Value: !Ref PublicSubnet2Cidr
  Description: ID of PublicSubnet2
  Export:
    Name: !Sub ${AWS::StackName}-PublicSubnet2Cidr
```

```
PrivateSubnet1Cidr:
  Value: !Ref PrivateSubnet1Cidr
  Description: ID of PublicSubnet1
  Export:
    Name: !Sub ${AWS::StackName}-PrivateSubnet1Cidr

PrivateSubnet2Cidr:
  Value: !Ref PublicSubnet2Cidr
  Description: ID of PrivateSubnet2Cidr
  Export:
    Name: !Sub ${AWS::StackName}-PrivateSubnet2Cidr


PublicRouteTable:
  Value: !Ref PublicRouteTable
  Description: ID of PublicRoute
  Export:
    Name: !Sub ${AWS::StackName}-PublicRouteTable

PrivateRouteTable:
  Value: !Ref PrivateRouteTable
  Description: ID of PrivateRouteTable
  Export:
    Name: !Sub ${AWS::StackName}-PrivateRouteTable

VPCDefaultSG:
  Value: !GetAtt LabVPC.DefaultSecurityGroup
  Description: ID of VPC default sg
  Export:
    Name: !Sub ${AWS::StackName}-VPCDefaultSG
```

[부록 10] RDS.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: RDS Template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values VPC
    Type: String
  DBUsername:
    Description: DB Username
    Default: admin
    NoEcho: true
    Type: String
    MinLength: '1'
    MaxLength: '16'
  DBPassword:
    Description: DB Pwd
    NoEcho: true
    Type: String
    MinLength: '8'
    MaxLength: '41'

  DBInstanceIdentifier:
    Description: Instance Identifier name
    Type: String

Resources:

############
# RDS
############
```

```
  RDSInstance:
    Type: AWS::RDS::DBInstance
    Properties:
      DBName: DBName
      Engine: MySQL
      EngineVersion: 8.0.20
      MasterUsername: !Ref DBUsername
      MasterUserPassword: !Ref DBPassword
      DBInstanceClass: db.t3.micro
      VPCSecurityGroups:
        -  Fn::GetAtt: [ RDSsg, GroupId ]
      AllocatedStorage: 20
      AvailabilityZone: !Select
        - 0
        - !GetAZs
          Ref: AWS::Region
      StorageType: gp2
      DBInstanceIdentifier: !Ref DBInstanceIdentifier
      PubliclyAccessible: false
      DBSubnetGroupName: !Ref RDSSubnetGroup

############
# RDS Subnet Group
############

  RDSSubnetGroup:
    Type: "AWS::RDS::DBSubnetGroup"
    Properties:
      DBSubnetGroupName: RDS Private Subnet Group
      DBSubnetGroupDescription: RDS Subnet Group
      SubnetIds:
        - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1'
```

```
        - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet2'
      Tags:
        -
          Key: Name
          Value: RDSSubnetGroup

############
# RDS sg
############

  RDSsg:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: RDSsg
      GroupDescription: Open Port 3306
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 3306
          ToPort: 3306
          CidrIp:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1Cidr'
        - IpProtocol: tcp
          FromPort: 3306
          ToPort: 3306
          CidrIp:
            Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1Cidr'
        - IpProtocol: tcp
          FromPort: 3306
          ToPort: 3306
          CidrIp:
            Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet2Cidr'

      Tags:
```

```
        -
          Key: Name
          Value: RDSsg
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'


Outputs:

  RDSEndPointAddress:
    Value: !GetAtt RDSInstance.Endpoint.Address
    Description: RDS EndPoint Address
    Export:
      Name: !Sub ${AWS::StackName}-RDSEndPointAddress
```

[부록 11] Bastion.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: Bastion Host template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  BastionAMIID:
```

```
    Type: String
    Default: ami-092faff259afb9a26




Resources:

############
# Bastion
##########

  BastionInstance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref BastionAMIID
      SubnetId:
        Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
      SecurityGroupIds:
        - !Ref BastionSecurityGroup
      IamInstanceProfile: !Ref InstanceProfile
      Tags:
        - Key: Name
          Value: Bastion Host
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          apt update
          apt install -y awscli
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
```

```
"awscliv2.zip"
          apt install unzip
          unzip awscliv2.zip
          sudo ./aws/install

          apt install -y mysql-client


############
# Bastion SG
############

  BastionSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: Bastion
      GroupDescription: Open Port 22 for ssh
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: Bastion


############
# IAM Role for Instance
############
```

```
JpBastionRole:
  Type: AWS::IAM::Role
  Properties:
    AssumeRolePolicyDocument:
      Version: 2012-10-17
      Statement:
        - Effect: Allow
          Principal:
            Service:
              - ec2.amazonaws.com
          Action:
            - sts:AssumeRole
    Path: /
    Policies:
      - PolicyName: JpBastionEc2FullAccess
        PolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Action: 'ec2:*'
              Resource: '*'
            - Effect: Allow
              Action: 'cloudformation:ListExports'
              Resource: '*'
            - Effect: Allow
              Action: 'cloudformation:DescribeStacks'
              Resource: '*'
            - Effect: Allow
              Action: 'rds:*'
              Resource: '*'
            - Effect: Allow
              Action: 's3:*'
              Resource: '*'
```

```
############
# Instance Profile
############

  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref JpBastionRole]
      InstanceProfileName: JpBastionRole



Outputs:
  BastionPublicDns:
    Value: !GetAtt BastionInstance.PublicDnsName
    Description: Bastion Dns Name
    Export:
      Name: !Sub ${AWS::StackName}-BastionPublicDns

  BastionId:
    Value: !Ref BastionInstance
    Description: Bastion Instance Id
    Export:
      Name: !Sub ${AWS::StackName}-BastionId

  BastionIp:
    Value: !GetAtt BastionInstance.PrivateIp
    Description: BastionInstance Instance ip
    Export:
      Name: !Sub ${AWS::StackName}-BastionInstanceIp
```

[부록 12] NAT.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: NAT Instance template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values VPCs
    Type: String

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  UbuntuAMI:
    Type: String
    Description: NAT Instance AMI ID
    Default: ami-092faff259afb9a26




Resources:

############
# NAT Instance
###########

  NatInstance:
```

```
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref UbuntuAMI
      SubnetId:
        Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
      SecurityGroupIds:
        - !Ref NATSecurityGroup
      IamInstanceProfile: !Ref InstanceProfile
      Tags:
        - Key: Name
          Value: NAT
      SourceDestCheck: False
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash

          sudo apt update -y
          sudo apt install -y awscli
          sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
          sudo apt install unzip
          sudo unzip awscliv2.zip
          sudo ./aws/install

          sudo apt-get update
          echo iptables-persistent iptables-persistent/autosave_v4 boolean true | sudo
debconf-set-selections
          echo iptables-persistent iptables-persistent/autosave_v6 boolean true | sudo
debconf-set-selections
          sudo apt-get -y install iptables-persistent
          echo 1 > /proc/sys/net/ipv4/ip_forward
```

```
          sudo sed -i '28s/#//g' /etc/sysctl.conf

          iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
          iptables -t nat -A POSTROUTING -o eth0 -s 0.0.0.0/0 -j MASQUERADE

          iptables -t nat -L
          netfilter-persistent save
          cat /etc/iptables/rules.v4


############
# NAT SG
############

  NATSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: Nat
      GroupDescription: Open Port 22 for ssh
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 0
          ToPort: 65535
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: NAT


############
```

```
# IAM Role for Instance
############

  jpNatRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: jpNatFullAccess
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'ec2:*'
                Resource: '*'


############
# Instance Profile
############

  InstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
```

```
      Path: /
      Roles: [!Ref jpNatRole]
      InstanceProfileName: jpNatRole



############
#PrivateRoute
############


  PrivateRoute:
    Type: AWS::EC2::Route
    Properties:
      DestinationCidrBlock: 0.0.0.0/0
      InstanceId: !Ref NatInstance
      RouteTableId:
        Fn::ImportValue: !Sub '${ExportStackName}-PrivateRouteTable'




Outputs:
  NATPublicDns:
    Value: !GetAtt NatInstance.PublicDnsName
    Description: NAT Dns Name
    Export:
      Name: !Sub ${AWS::StackName}-NATDns

  NATId:
    Value: !Ref NatInstance
```

```
    Description: NAT Instance Id
    Export:
      Name: !Sub ${AWS::StackName}-NATId

  NATIp:
    Value: !GetAtt NatInstance.PrivateIp
    Description: NAT Instance ip
    Export:
      Name: !Sub ${AWS::StackName}-NATIp
```

[부록 13] Golden_AMI.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: GoldenAMI template


Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  GoldenAMIID:
    Type: String
    Description: Insert ami id of ubuntu 18.04
    Default: ami-092faff259afb9a26
```

```
Resources:

############
# GoldenAMI
##########

  GoldenAMI:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref GoldenAMIID
      SubnetId:
        Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1'
      SecurityGroupIds:
        - !Ref GoldenAMISecurityGroup
      IamInstanceProfile: !Ref InstanceProfile
      Tags:
        - Key: Name
          Value: Golden AMI
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash
          apt update
          apt install -y awscli
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
          apt install unzip
          unzip awscliv2.zip
          sudo ./aws/install
          sudo apt install -y docker.io
          sudo systemctl start docker
          aws s3 cp s3://golden-ami-suyeonshoes/aws-sam-cli-linux-x86_64.zip ./
```

```
        sudo unzip aws-sam-cli-linux-x86_64.zip -d sam-installation
        sudo ./sam-installation/install

        aws s3 cp s3://golden-ami-suyeonshoes2/AS_Region2.zip ./
        sudo unzip AS_Region2.zip
        sam build --use-container
        sam package --output-template-file packaged.yaml --s3-bucket golden-ami-suyeonshoes2 --region ap-northeast-3
        sam deploy --template-file packaged.yaml --stack-name myAS --capabilities CAPABILITY_IAM --region ap-northeast-3


############
# Golden AMI SG
############

  GoldenAMISecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: GoldenAMI
      GroupDescription: Open Port 22 for ssh
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1Cidr'
      Tags:
        - Key: Name
          Value: GoldenAMISecurityGroup
```

```
############
# IAM Role for Instance
############

  jpGoldenAMIRole:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: jpGoldenAMIEc2FullAccess
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'ec2:*'
                Resource: '*'
              - Effect: Allow
                Action: 'cloudformation:*'
                Resource: '*'
              - Effect: Allow
                Action: 'iam:*'
                Resource: '*'
              - Effect: Allow
```

```
            Action: 's3:*'
            Resource: '*'
          - Effect: Allow
            Action: 'imagebuilder:*'
            Resource: '*'
          - Effect: Allow
            Action: 'ssm:*'
            Resource: '*'
          - Effect: Allow
            Action: 'lambda:*'
            Resource: '*'
          - Effect: Allow
            Action: 'autoscaling:*'
            Resource: '*'
          - Effect: Allow
            Action: 'sts:*'
            Resource: '*'
          - Effect: Allow
            Action: 'sns:*'
            Resource: '*'
          - Effect: Allow
            Action: 'elasticloadbalancing:*'
            Resource: '*'
          - Effect: Allow
            Action: 'rds:*'
            Resource: '*'


###########
# Instance Profile
###########

  InstanceProfile:
```

```
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref jpGoldenAMIRole]
      InstanceProfileName: jpGoldenAMIRole



Outputs:
  GoldenAMIaddr:
    Value: !GetAtt GoldenAMI.PrivateIp
    Description: GoldenAMIaddr
    Export:
      Name: !Sub ${AWS::StackName}-GoldenAMIaddr

  GoldenAMIId:
    Value: !Ref GoldenAMI
    Description: GoldenAMI Instance Id
    Export:
      Name: !Sub ${AWS::StackName}-GoldenAMIId
```

[부록 14] template.yaml(Auto Scaling, EC2 Image Builder, SNS, Lambda)

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'

Metadata:
  License:
    Description: >
      Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
      SPDX-License-Identifier: MIT-0
      AutoScalingGroupInstanceType
```

```
Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String
    Default: "myVPC"
  EnvironmentName:
    Type: String
    Default: "Web Server"
  AmiId:
    Type: String
    Default: "ami-07763aa528e05b51e"
  BuildInstanceType:
    Type: String
```

```
    Default: "t2.micro"
    Description: "Image Builder instance type"
  AutoScalingGroupInstanceType:
    Type: String
    Default: "t2.micro"
    Description: Instance type for sample Auto Scaling groupi
  KeyName:
    Type: String
    Default: "jpAdmin"
  Email:
    Type: String
    Default: "kofdx7@gmail.com"

Resources:


############
# InstanceRefreshHandler
############
  InstanceRefreshHandler:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: lambda_function.lambda_handler
      Runtime: python3.7
      MemorySize: 128
      Timeout: 30
      Role: !GetAtt InstanceRefreshHandlerLambdaRole.Arn
      CodeUri: InstanceRefreshHandler/
      Environment:
        Variables:
          AutoScalingGroupName: !Ref AutoScalingGroup
```

```
############
# InstanceRefreshHandlerLambdaRole
############
  InstanceRefreshHandlerLambdaRole:
    Type: 'AWS::IAM::Role'
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
        - Effect: Allow
          Principal:
            Service:
            - lambda.amazonaws.com
          Action:
          - sts:AssumeRole
      Path: "/service-role/"
      Policies:
      - PolicyName: jplambdaExecution-InstanceRefreshHandler
        PolicyDocument:
          Version: '2012-10-17'
          Statement:
          - Effect: Allow
            Action:
            - logs:CreateLogGroup
            Resource: '*'
          - Effect: Allow
            Action:
            - logs:CreateLogStream
            - logs:PutLogEvents
            Resource: '*'
          - Effect: Allow
            Action:
            - autoscaling:StartInstanceRefresh
```

```
            - autoscaling:Describe*
            - ec2:CreateLaunchTemplateVersion
            - ec2:DescribeLaunchTemplates
            Resource: '*'

############
# ImageBuilderSNSTopic
############
  ImageBuilderSNSTopic:
    Type: "AWS::SNS::Topic"
    Properties:
      Subscription:
        - Endpoint: !GetAtt InstanceRefreshHandler.Arn
          Protocol: lambda
############
# SNSLambdaPermission
############

  SNSLambdaPermission:
    Type: AWS::Lambda::Permission
    Properties:
        FunctionName: !GetAtt InstanceRefreshHandler.Arn
        Action: lambda:InvokeFunction
        Principal: sns.amazonaws.com
        SourceArn: !Ref ImageBuilderSNSTopic


############
# EC2ImageBuilderRecipe
############
  EC2ImageBuilderRecipe:
    Type: AWS::ImageBuilder::ImageRecipe
    Properties:
```

```
      Name: EC2ImageBuilderRecipe
      Description: This recipe updates the system and installs Docker CE
      ParentImage: !Ref AmiId
      Components:
        - ComponentArn: !Sub
"arn:aws:imagebuilder:${AWS::Region}:aws:component/update-linux/1.0.0/1"
        - ComponentArn: !Sub
"arn:aws:imagebuilder:${AWS::Region}:aws:component/reboot-linux/1.0.1/1"
      Version: "1.0.0"

############
# EC2ImageBuilderPipeline
############
  EC2ImageBuilderPipeline:
    Type: AWS::ImageBuilder::ImagePipeline
    Properties:
      Name: EC2ImageBuilderPipeline
      ImageRecipeArn: !Ref EC2ImageBuilderRecipe
      InfrastructureConfigurationArn: !Ref EC2ImageBuilderInfrastructureConfiguration
      DistributionConfigurationArn: !Ref DistributionConfigurationArn


############
# EC2ImageBuilder DistributionConfiguration
############
  DistributionConfiguration:
    Type: AWS::ImageBuilder::DistributionConfiguration
    Properties:
      Name: 'distribution-configuration-name'
      Description: 'description'
      Distributions:
        - Region: 'ap-northeast-3'
          AmiDistributionConfiguration:
```

```
            Name: 'distribution japan {{ imagebuilder:buildDate }}'
        - Region: 'us-east-1'
          AmiDistributionConfiguration:
            Name: 'distribution backup {{ imagebuilder:buildDate }}'
      Tags:
        CustomerDistributionConfigTagKey1: 'CustomerDistributionConfigTagValue1'
        CustomerDistributionConfigTagKey2: 'CustomerDistributionConfigTagValue2'




###########
# EC2ImageBuilderInfrastructureConfiguration
###########
  EC2ImageBuilderInfrastructureConfiguration:
    Type: AWS::ImageBuilder::InfrastructureConfiguration
    Properties:
      Name: InstanceConfigurationForEC2ImageBuilder
      InstanceTypes:
        - !Ref BuildInstanceType
      InstanceProfileName: !Ref EC2ImageBuilderIAMInstanceProfile
      SNSTopicArn: !Ref ImageBuilderSNSTopic
      SubnetId:
         Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
      SecurityGroupIds:
        - !Ref ASTemplateGroup
      TerminateInstanceOnFailure: true

###########
# EC2ImageBuilderIAMRole
###########
  EC2ImageBuilderIAMRole:
    Type: AWS::IAM::Role
```

```
    Properties:
      AssumeRolePolicyDocument:
        Version: "2012-10-17"
        Statement:
          - Effect: "Allow"
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/EC2InstanceProfileForImageBuilder
        - arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore
      Policies:
        - PolicyName: jpEC2ImageBuilderIAM
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'cloudformation:ListExports'
                Resource: '*'
              - Effect: Allow
                Action: 'cloudformation:DescribeStacks'
                Resource: '*'


###########
# EC2ImageBuilderIAMInstanceProfile
###########
  EC2ImageBuilderIAMInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Roles:
```

```
        - !Ref EC2ImageBuilderIAMRole


############
# AutoScaling Group
############
  AutoScalingGroup:
    Type: AWS::AutoScaling::AutoScalingGroup
    Properties:
      MinSize: "1"
      MaxSize: "4"
      DesiredCapacity: "2"
      HealthCheckGracePeriod: 180
      HealthCheckType: ELB
      TargetGroupARNs:
        - !Ref LoadBalancerTargetGroup
      VPCZoneIdentifier:
        - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet1'
        - Fn::ImportValue: !Sub '${ExportStackName}-PrivateSubnet2'
      MetricsCollection:
        - Granularity: "1Minute"
      CapacityRebalance: true
      NotificationConfigurations:
      - TopicARN: !Ref SnsTopic
        NotificationTypes:
        - autoscaling:EC2_INSTANCE_LAUNCH
        - autoscaling:EC2_INSTANCE_LAUNCH_ERROR
        - autoscaling:EC2_INSTANCE_TERMINATE
        - autoscaling:EC2_INSTANCE_TERMINATE_ERROR
        - autoscaling:TEST_NOTIFICATION
      LaunchTemplate:
        LaunchTemplateId: !Ref LaunchTemplate
        Version: !GetAtt LaunchTemplate.LatestVersionNumber
```

```
      Tags:
        - Key: Name
          Value: !Ref EnvironmentName
          PropagateAtLaunch: true

############
# AS-CPU-Policy
############
  myCPUPolicy:
    Type: AWS::AutoScaling::ScalingPolicy
    Properties:
      AutoScalingGroupName: !Ref AutoScalingGroup
      PolicyType: TargetTrackingScaling
      TargetTrackingConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ASGAverageCPUUtilization
        TargetValue: 60.00


############
# SNS-Topic
############
  SnsTopic:
    Type: AWS::SNS::Topic
    Properties:
      DisplayName: topic-sns
      FifoTopic: False
      TopicName: Project-sns

############
# SNS-Subscription
############
  SnsSub:
```

```
    Type: AWS::SNS::Subscription
    Properties:
      Endpoint: !Ref Email
      Protocol: Email
      TopicArn: !Ref SnsTopic


  LaunchTemplate:
    Type: AWS::EC2::LaunchTemplate
    Properties:
      LaunchTemplateData:
        ImageId: !Ref AmiId
        KeyName: !Ref KeyName
        IamInstanceProfile:
          Arn: !GetAtt EC2ImageBuilderIAMInstanceProfile.Arn
        InstanceType: !Ref AutoScalingGroupInstanceType
        SecurityGroupIds:
          - !Ref ASTemplateGroup
        UserData:
          Fn::Base64: !Sub |
            #!/bin/bash
            sudo apt update -y
            sudo apt install -y awscli
            sudo curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
            sudo apt install unzip
            sudo unzip awscliv2.zip
            sudo ./aws/install

            dbaddr='$mysql_hostname='"'"`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`RDSEndPointAddress₩`].OutputValue' --output=text
--region=ap-northeast-3`"';"
            sudo sed -i "4s/.*/$dbaddr/g" /var/www/html/basic/login/dbconn.php
```

```
############
# Application Load Balancer
############
  LoadBalancer:
    Type: AWS::ElasticLoadBalancingV2::LoadBalancer
    Properties:
      Type: application
      Name: WebServerALB
      Scheme: internet-facing
      SubnetMappings:
        - SubnetId:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet1'
        - SubnetId:
            Fn::ImportValue: !Sub '${ExportStackName}-PublicSubnet2'
      IpAddressType: ipv4
      SecurityGroups:
        - !Ref ASTemplateGroup

############
# Application Load Balancer TargetGroup
############
  LoadBalancerTargetGroup:
    Type: AWS::ElasticLoadBalancingV2::TargetGroup
    Properties:
      Name: ALB-Target
      TargetType: instance
      HealthyThresholdCount: 2
      HealthCheckIntervalSeconds: 30
      TargetGroupAttributes:
        - Key: stickiness.enabled
          Value: true
        - Key: stickiness.type
```

```
          Value: lb_cookie
        - Key: stickiness.lb_cookie.duration_seconds
          Value: 86400
      Protocol: HTTP
      Port: 80
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'


###########
# Application Load Balancer Listener
###########
  LoadBalancerListener:
   Type: AWS::ElasticLoadBalancingV2::Listener
   Properties:
     DefaultActions:
       - Type: "forward"
         ForwardConfig:
           TargetGroups:
            - TargetGroupArn: !Ref LoadBalancerTargetGroup
     LoadBalancerArn: !Ref LoadBalancer
     Port: 80
     Protocol: "HTTP"


###########
# Auto Scaling Temp Security Group
###########

  ASTemplateGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: ASTemplate
```

```
      GroupDescription: ASG Template Security Group
      VpcId:
        Fn::ImportValue: !Sub '${ExportStackName}-LabVPC'
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 80
          ToPort: 80
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: ASTemplate




Outputs:
  EC2ImageBuilderPipeline:
    Description: Sample EC2 Image Builder Pipeline
    Value: !Ref EC2ImageBuilderPipeline
  SNSTopic:
    Description: Amazon SNS topic subscribed to the EC2 Image Builder pipeline to
trigger Lambda
    Value: !Ref ImageBuilderSNSTopic
  LambdaFunction:
    Description: AWS Lambda function handling EC2 Image Builder Notifications and
triggering Auto Scaling Instance Refresh
    Value: !Ref InstanceRefreshHandler
  AutoScalingGroup:
    Description: Sample Auto Scaling group
    Value: !Ref AutoScalingGroup
  LaunchTemplate:
    Description: Sample Launch Template for Auto Scaling group
    Value: !Ref LaunchTemplate
```

```
  ALBDns:
    Value: !GetAtt LoadBalancer.DNSName
    Description: Load Balancer Dns
    Export:
      Name: !Sub ${AWS::StackName}-LoadBalancerDns
  ALBHz:
    Value: !GetAtt LoadBalancer.CanonicalHostedZoneID
    Description: Load Balancer HZ
    Export:
      Name: !Sub ${AWS::StackName}-LoadBalancerHZ
```

[부록 15] Route53.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: Route53 template

Parameters:
  ExportStackName:
    Description: The name of the stack that exports the values
    Type: String

  JpALBDns:
    Description: The name of the stack that exports the values
    Type: String

  JpALBHz:
    Description: The name of the stack that exports the values
    Type: String
  HzId:
    Description: The name of the stack that exports the values
```

```
    Type: String
    Default: Z02045551T74ATR5QH0EX


Resources:

  Server:
    Type: AWS::Route53::RecordSetGroup
    Properties:
      Comment: Register DNS
      HostedZoneId:
        !Ref HzId
      RecordSets:
      - Name: jp.suyeonshoes.click
        Type: A
        AliasTarget:
          HostedZoneId:
            !Ref JpALBHz
          DNSName:
            !Ref JpALBDns
```

[부록 16] lambda.py

```
import boto3
from botocore.exceptions import ClientError
import json
import logging
import os

logger = logging.getLogger()
logger.setLevel(logging.INFO)

# define boto3 clients
ec2_client = boto3.client('ec2')
```

```
asg_client = boto3.client('autoscaling')


def get_ami_id_from_ib_notification(ib_notification):
    # Parse Image Builder notification and look up AMI for Lambda region
    for resource in ib_notification['outputResources']['amis']:
        if resource['region'] == os.environ['AWS_REGION']:
            return(resource['image'])
        else:
            return(None)


def get_launch_template_id_for_asg(asg_name):
    try:
        asg_describe = asg_client.describe_auto_scaling_groups(
            AutoScalingGroupNames=[asg_name])

        # Make sure the Auto Scaling group exists
        if len(asg_describe['AutoScalingGroups']) == 0:
            raise ValueError(
                "The configured Auto Scaling group "
                "does not exist: {}".format(asg_name))

        asg_details = asg_describe['AutoScalingGroups'][0]

        # ASG may have a LaunchTemplate, MixedInstancePolicy or
LaunchConfiguration
        if 'LaunchTemplate' in asg_details.keys():
            return(asg_details['LaunchTemplate']['LaunchTemplateId'])
        elif 'MixedInstancesPolicy' in asg_details.keys():
            return(asg_details['MixedInstancesPolicy']
                              ['LaunchTemplate']
                              ['LaunchTemplateSpecification']
```

```
                                ['LaunchTemplateId'])
        else:
            return(None)

    except ClientError as e:
        logging.error("Error describing Auto Scaling group.")
        raise e


def create_launch_template_version_with_new_ami(lt_id, ami_id):
    try:
        latest_lt_version = ec2_client.describe_launch_templates(
            LaunchTemplateIds=[lt_id])['LaunchTemplates'][0]['LatestVersionNumber']

        response = ec2_client.create_launch_template_version(
                            LaunchTemplateId=lt_id,
                            SourceVersion=str(latest_lt_version),
                            LaunchTemplateData={'ImageId': ami_id})
        logging.info("Created new launch template version for {} : {} with "
                     "image {}".format(lt_id, str(latest_lt_version), ami_id))
        return(response)

    except ClientError as e:
        logging.error('Error creating the new launch template version')
        raise e


def trigger_auto_scaling_instance_refresh(asg_name, strategy="Rolling",
                                          min_healthy_percentage=90, instance_warmup=300):

    try:
        response = asg_client.start_instance_refresh(
            AutoScalingGroupName=asg_name,
```

```
            Strategy=strategy,
            Preferences={
                'MinHealthyPercentage': min_healthy_percentage,
                'InstanceWarmup': instance_warmup
            })
        logging.info("Triggered Instance Refresh {} for Auto Scaling "
                     "group {}".format(response['InstanceRefreshId'], asg_name))

    except ClientError as e:
        logging.error("Unable to trigger Instance Refresh for "
                      "Auto Scaling group {}".format(asg_name))
        raise e


def lambda_handler(event, context):

    # Load SNS message body
    ib_notification = json.loads(event['Records'][0]['Sns']['Message'])

    asg_name = os.environ['AutoScalingGroupName']

    # Finish if Image build wasn't successful
    if ib_notification['state']['status'] != "AVAILABLE":
        logging.warning("No action taken. EC2 Image build failed.")
        return("No action taken. EC2 Image build failed.")

    # Get the AMI ID for current region from Image Builder notification
    ami_id = get_ami_id_from_ib_notification(ib_notification)
    if ami_id is None:
        logging.warning("There's no image created for region {}".format(
                        os.environ['AWS_REGION']))
        return("No AMI id created for region {}".format(
                os.environ['AWS_REGION']))
```

```
    # Get Launch Template id for the Auto Scaling group
    lt_id = get_launch_template_id_for_asg(asg_name)
    if lt_id is None:
        raise ValueError("Auto Scaling group {} doesn't use a "
                         "Launch Template".format(asg_name))

    # Get latest version for Launch template to use as source and create
    # new version with new AMI
    create_launch_template_version_with_new_ami(lt_id, ami_id)

    # Trigger Auto Scaling group Instance Refresh
    trigger_auto_scaling_instance_refresh(asg_name)

    return("Success")
```

[부록 17] DR_Region1.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: this template is for DR of region1, region2


Parameters:

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  UbuntuAMIID:
    Type: AWS::SSM::Parameter::Value<AWS::EC2::Image::Id>
```

```
    Default:
/aws/service/canonical/ubuntu/server/18.04/stable/20210224/amd64/hvm/ebs-
gp2/ami-id

Resources:

############
# DR Instance
############

  DRInstance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref UbuntuAMIID
      SecurityGroupIds:
        - !Ref DRSecurityGroup
      IamInstanceProfile: !Ref DRInstanceProfile
      Tags:
        - Key: Name
          Value: DR
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash

          ############################
          #install aws cli2
          ############################
          apt update
          apt install -y awscli
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
```

```
apt install unzip
unzip awscliv2.zip
sudo ./aws/install

##########################
#Config to Korea Seoul Region /ap-northeast-2/
##########################

##########################
#create VPC stack on ap-northeast-2
##########################
aws cloudformation create-stack --stack-name myVPC --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region1/1.VPC.yaml --
parameters ParameterKey=LabVpcCidr,ParameterValue=10.5.0.0/16
ParameterKey=PublicSubnet1Cidr,ParameterValue=10.5.10.0/24
ParameterKey=PublicSubnet2Cidr,ParameterValue=10.5.30.0/24
ParameterKey=PrivateSubnet1Cidr,ParameterValue=10.5.20.0/24
ParameterKey=PrivateSubnet2Cidr,ParameterValue=10.5.40.0/24  --region ap-
northeast-2

##########################
#wait vpc stack
##########################
aws cloudformation wait stack-create-complete --stack-name myVPC --region
ap-northeast-2

##########################
#create rds stack on ap-northeast-2
##########################
aws cloudformation create-stack --stack-name myRDS --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region1/2.RDS_template.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=DBUsername,ParameterValue=admin
```

```
ParameterKey=DBPassword,ParameterValue=123456789
ParameterKey=DBInstanceIdentifier,ParameterValue=dbinstance-ap-northeast-2 --
region ap-northeast-2

        ############################
        #wait rds stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myRDS --region
ap-northeast-2

        ############################
        #create bastion stack on ap-northeast-2
        ############################
        aws cloudformation create-stack --stack-name myBastion --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region1/3.Bastion.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KeyName,ParameterValue=krAdmin
ParameterKey=BastionAMIID,ParameterValue=/aws/service/canonical/ubuntu/server/1
8.04/stable/20210224/amd64/hvm/ebs-gp2/ami-id --region ap-northeast-2 --
capabilities CAPABILITY_NAMED_IAM

        ############################
        #wait bastion stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myBastion --
region ap-northeast-2

        ############################
        #KrBastionId is Output value of Bastion's stack
        ############################
        KrBastionId=`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`BastionId₩`].OutputValue' --output=text --
region=ap-northeast-2`
```

```
        ############################
        #wait bastion instance's status
        ############################
        aws ec2 wait instance-status-ok --instance-ids $KrBastionId --region ap-
northeast-2

        ############################
        #create nat instance stack on ap-northeast-2
        ############################
        aws cloudformation create-stack --stack-name myNAT --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region1/4.NAT.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KeyName,ParameterValue=krAdmin
ParameterKey=UbuntuAMI,ParameterValue=ami-0ba5cd124d7a79612 --region ap-
northeast-2 --capabilities CAPABILITY_NAMED_IAM

        ############################
        #wait nat instance stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myNAT --
region ap-northeast-2

        ############################
        #create GoldenAMI instance stack on ap-northeast-2
        ############################
        aws cloudformation create-stack --stack-name myGoldenAMI --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region1/5.Golden_AMI.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KeyName,ParameterValue=krAdmin
ParameterKey=GoldenAMIID,ParameterValue=ami-0c3875b921d298b30 --region ap-
northeast-2 --capabilities CAPABILITY_NAMED_IAM
```

```
        ############################
        #wait GoldenAMI stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myGoldenAMI --region ap-northeast-2

        ############################
        #HzId is HostZoneId of suyeonshoes.click
        ############################
        HzId=`aws route53 list-hosted-zones-by-name --dns-name "suyeonshoes.click." --query HostedZones[].Id --output=text`

        ############################
        #GoldenAMIId is Output value of GoldenAMIId's stack
        ############################
        GoldenAMIId=`aws cloudformation describe-stacks --query 'Stacks[].Outputs[?OutputKey==₩`GoldenAMIId₩`].OutputValue' --output=text --region=ap-northeast-2`

        ############################
        #wait GoldenAMIId instance's status
        ############################
        aws ec2 wait instance-status-ok --instance-ids $GoldenAMIId --region ap-northeast-2

        ############################
        #wait myAS stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myAS --region ap-northeast-2

        ############################
        #KrALBDNs and KrALBHz is output value of ap-norhteast-2's myAS stack
```

```
        ############################
        KrALBDns=`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`ALBDns₩`].OutputValue' --output=text --
region=ap-northeast-2`
        KrALBHz=`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`ALBHz₩`].OutputValue' --output=text --region=ap-
northeast-2`

        ############################
        #create Route stack on ap-northeast-2
        ############################
        aws cloudformation create-stack --stack-name myRoute53 --template-url
https://drtemplate.s3.ap-northeast-
2.amazonaws.com/Region1/6.Route53_Region1.yaml --parameters
ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KrALBDns,ParameterValue=$KrALBDns
ParameterKey=KrALBHz,ParameterValue=$KrALBHz
ParameterKey=HzId,ParameterValue=$HzId --region ap-northeast-2 --capabilities
CAPABILITY_NAMED_IAM



############
# DRSecurityGroup
############

  DRSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: DRSecurityGroup
      GroupDescription: Open Port 22 , 80 for ssh
      SecurityGroupIngress:
        - IpProtocol: tcp
```

```
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
      Tags:
        - Key: Name
          Value: DRSecurityGroup


###########
# IAM Role for Instance
###########

  krDRRoles:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: FullAccess
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'rds:*'
                Resource: '*'
```

```
    - Effect: Allow
      Action: 'ec2:*'
      Resource: '*'
    - Effect: Allow
      Action: 's3:*'
      Resource: '*'
    - Effect: Allow
      Action: 'cloudformation:*'
      Resource: '*'
    - Effect: Allow
      Action: 'ssm:*'
      Resource: '*'
    - Effect: Allow
      Action: 'elasticloadbalancing:*'
      Resource: '*'
    - Effect: Allow
      Action: 'iam:*'
      Resource: '*'
    - Effect: Allow
      Action: 'autoscaling:*'
      Resource: '*'
    - Effect: Allow
      Action: 'route53:*'
      Resource: '*'
    - Effect: Allow
      Action: 'sns:*'
      Resource: '*'
    - Effect: Allow
      Action: 'sts:*'
      Resource: '*'
    - Effect: Allow
      Action: 'imagebuilder:*'
      Resource: '*'
```

```
              - Effect: Allow
                Action: 'lambda:*'
                Resource: '*'
###########
# Instance Profile
###########

  DRInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref krDRRoles]
      InstanceProfileName: krDRRoles
```

[부록 18] DR_Region2.yaml

```
AWSTemplateFormatVersion: 2010-09-09
Description: this template is for DR of region1, region2


Parameters:

  KeyName:
    Type: "AWS::EC2::KeyPair::KeyName"
    Description: Select KeyPair

  UbuntuAMIID:
    Type: String
```

```
    Default: ami-092faff259afb9a26

Resources:

############
# DR Instance
############

  DRInstance:
    Type: AWS::EC2::Instance
    Properties:
      KeyName: !Ref KeyName
      InstanceType: t2.micro
      ImageId: !Ref UbuntuAMIID
      SecurityGroupIds:
        - !Ref DRSecurityGroup
      IamInstanceProfile: !Ref DRInstanceProfile
      Tags:
        - Key: Name
          Value: DR
      UserData:
        Fn::Base64: !Sub |
          #!/bin/bash

          ############################
          #install aws cli2
          ############################
          apt update
          apt install -y awscli
          curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
          apt install unzip
          unzip awscliv2.zip
```

```
sudo ./aws/install

##########################
#Config to Japan Osaka Region /ap-northeast-3/
##########################

##########################
#create VPC stack on ap-northeast-3
##########################
aws cloudformation create-stack --stack-name myVPC --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region2/1.VPC.yaml --
parameters ParameterKey=LabVpcCidr,ParameterValue=192.168.0.0/16
ParameterKey=PublicSubnet1Cidr,ParameterValue=192.168.10.0/24
ParameterKey=PublicSubnet2Cidr,ParameterValue=192.168.30.0/24
ParameterKey=PrivateSubnet1Cidr,ParameterValue=192.168.20.0/24
ParameterKey=PrivateSubnet2Cidr,ParameterValue=192.168.40.0/24  --region ap-
northeast-3

##########################
#wait vpc stack
##########################
aws cloudformation wait stack-create-complete --stack-name myVPC --region
ap-northeast-3

##########################
#create rds stack on ap-northeast-3
##########################
aws cloudformation create-stack --stack-name myRDS --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region2/2.RDS_template.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=DBUsername,ParameterValue=admin
ParameterKey=DBPassword,ParameterValue=123456789
ParameterKey=DBInstanceIdentifier,ParameterValue=dbinstance-ap-northeast-3 --
```

```
region ap-northeast-3

        ############################
        #wait rds stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myRDS --region
ap-northeast-3

        ############################
        #create bastion stack on ap-northeast-2
        ############################
        aws cloudformation create-stack --stack-name myBastion --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region2/3.Bastion.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KeyName,ParameterValue=jpAdmin
ParameterKey=BastionAMIID,ParameterValue=ami-092faff259afb9a26 --region ap-
northeast-3 --capabilities CAPABILITY_NAMED_IAM

        ############################
        #wait bastion stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myBastion --
region ap-northeast-3

        ############################
        #JpBastionId is Output value of Bastion's stack
        ############################
        JpBastionId=`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`BastionId₩`].OutputValue' --output=text --
region=ap-northeast-3`

        ############################
        #wait bastion instance's status
```

```
        ############################
        aws ec2 wait instance-status-ok --instance-ids $JpBastionId --region ap-
northeast-3

        ############################
        #create nat instance stack on ap-northeast-3
        ############################
        aws cloudformation create-stack --stack-name myNAT --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region2/4.NAT.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KeyName,ParameterValue=jpAdmin
ParameterKey=UbuntuAMI,ParameterValue=ami-092faff259afb9a26 --region ap-
northeast-3 --capabilities CAPABILITY_NAMED_IAM

        ############################
        #wait nat instance stack
        ############################
        aws cloudformation wait stack-create-complete --stack-name myNAT --
region ap-northeast-3

        ############################
        #create GoldenAMI instance stack on ap-northeast-3
        ############################
        aws cloudformation create-stack --stack-name myGoldenAMI --template-url
https://drtemplate.s3.ap-northeast-2.amazonaws.com/Region2/5.Golden_AMI.yaml --
parameters ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=KeyName,ParameterValue=jpAdmin
ParameterKey=GoldenAMIID,ParameterValue=ami-092faff259afb9a26 --region ap-
northeast-3 --capabilities CAPABILITY_NAMED_IAM

        ############################
        #wait GoldenAMI stack
        ############################
```

```
aws cloudformation wait stack-create-complete --stack-name myGoldenAMI --region ap-northeast-3

############################
#HzId is HostZoneId of suyeonshoes.click
############################
HzId=`aws route53 list-hosted-zones-by-name --dns-name "suyeonshoes.click." --query HostedZones[].Id --output=text`

############################
#GoldenAMIId is Output value of GoldenAMIId's stack
############################
GoldenAMIId=`aws cloudformation describe-stacks --query 'Stacks[].Outputs[?OutputKey==₩`GoldenAMIId₩`].OutputValue' --output=text --region=ap-northeast-3`

############################
#wait GoldenAMIId instance's status
############################
aws ec2 wait instance-status-ok --instance-ids $GoldenAMIId --region ap-northeast-3

############################
#wait myAS stack
############################
aws cloudformation wait stack-create-complete --stack-name myAS --region ap-northeast-3


############################
#JpALBDNs and JpALBHz is output value of ap-norhteast-3's myAS stack
############################
JpALBDns=`aws cloudformation describe-stacks --query
```

```
'Stacks[].Outputs[?OutputKey==₩`ALBDns₩`].OutputValue' --output=text --
region=ap-northeast-3`
        JpALBHz=`aws cloudformation describe-stacks --query
'Stacks[].Outputs[?OutputKey==₩`ALBHz₩`].OutputValue' --output=text --region=ap-
northeast-3`

        ############################
        #create Route53 instance stack on ap-northeast-3
        ############################
        aws cloudformation create-stack --stack-name myRoute --template-url
https://drtemplate.s3.ap-northeast-
2.amazonaws.com/Region2/6.Route53_Region2.yaml --parameters
ParameterKey=ExportStackName,ParameterValue=myVPC
ParameterKey=JpALBDns,ParameterValue=$JpALBDns
ParameterKey=JpALBHz,ParameterValue=$JpALBHz
ParameterKey=HzId,ParameterValue=$HzId --region ap-northeast-3 --capabilities
CAPABILITY_NAMED_IAM


############
# DRSecurityGroup
############

  DRSecurityGroup:
    Type: AWS::EC2::SecurityGroup
    Properties:
      GroupName: DRSecurityGroup
      GroupDescription: Open Port 22 , 80 for ssh
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: 0.0.0.0/0
```

```
      Tags:
        - Key: Name
          Value: DRSecurityGroup


############
# IAM Role for Instance
############

  jpDRRoles:
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: 2012-10-17
        Statement:
          - Effect: Allow
            Principal:
              Service:
                - ec2.amazonaws.com
            Action:
              - sts:AssumeRole
      Path: /
      Policies:
        - PolicyName: FullAccess
          PolicyDocument:
            Version: "2012-10-17"
            Statement:
              - Effect: Allow
                Action: 'rds:*'
                Resource: '*'
              - Effect: Allow
                Action: 'ec2:*'
                Resource: '*'
```

```
    - Effect: Allow
      Action: 's3:*'
      Resource: '*'
    - Effect: Allow
      Action: 'cloudformation:*'
      Resource: '*'
    - Effect: Allow
      Action: 'ssm:*'
      Resource: '*'
    - Effect: Allow
      Action: 'elasticloadbalancing:*'
      Resource: '*'
    - Effect: Allow
      Action: 'iam:*'
      Resource: '*'
    - Effect: Allow
      Action: 'autoscaling:*'
      Resource: '*'
    - Effect: Allow
      Action: 'route53:*'
      Resource: '*'
    - Effect: Allow
      Action: 'sns:*'
      Resource: '*'
    - Effect: Allow
      Action: 'sts:*'
      Resource: '*'
    - Effect: Allow
      Action: 'imagebuilder:*'
      Resource: '*'
    - Effect: Allow
      Action: 'lambda:*'
      Resource: '*'
```

```
############
# Instance Profile
############

  DRInstanceProfile:
    Type: AWS::IAM::InstanceProfile
    Properties:
      Path: /
      Roles: [!Ref jpDRRoles]
      InstanceProfileName: jpDRRoles
```

[부록 19] GitHub

Cloud Formation Template Code